PREDICTIVE PERFORMANCE OF MACHINE LEARNING ALGORITHMS FOR ORE RESERVE ESTIMATION IN SPARSE AND IMPRECISE DATA

A

THESIS

Presented to the Faculty
of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

By

Sridhar Dutta, M. Tech

Fairbanks, Alaska

August 2006

PREDICTIVE PERFORMANCE

OF MACHINE LEARNING ALGORITHMS

FOR ORE RESERVE ESTIMATION IN SPARSE AND IMPRECISE DATA

By

Sridhar Dutta

RECOMMEND: [illegible]

Debasmita Misra

[illegible]

Sukumar Bandopadhyay
Advisory Committee Chair

[illegible]
Department Chair, Mining & Geological Engineering

APPROVED: [illegible]
Dean, College of Engineering and Mines

Susan M. Henrichs
Dean of Graduate School

July 20, 2006
Date

ABSTRACT

Traditional geostatistical estimation techniques have been used predominantly in the mining industry for the purpose of ore reserve estimation. Determination of mineral reserve has always posed considerable challenge to mining engineers due to geological complexities that are generally associated with the phenomenon of ore body formation. Considerable research over the years has resulted in the development of a number of state-of-the-art methods for the task of predictive spatial mapping such as ore reserve estimation. Recent advances in the use of the machine learning algorithms (MLA) have provided a new approach to solve the age-old problem. Therefore, this thesis is focused on the use of two MLA, viz. the neural network (NN) and support vector machine (SVM), for the purpose of ore reserve estimation. Application of the MLA have been elaborated with two complex drill hole datasets. The first dataset is a placer gold drill hole data characterized by high degree of spatial variability, sparseness and noise while the second dataset is obtained from a continuous lode deposit.

The application and success of the models developed using these MLA for the purpose of ore reserve estimation depends to a large extent on the data subsets on which they are trained and subsequently on the selection of the appropriate model parameters. The model data subsets obtained by random data division are not desirable in sparse data conditions as it usually results in statistically dissimilar subsets, thereby reducing their applicability. Therefore, an ideal technique for data subdivision has been suggested in the thesis. Additionally, issues pertaining to the optimum model development have also been discussed.

To investigate the accuracy and the applicability of the MLA for ore reserve estimation, their generalization ability was compared with the geostatistical ordinary kriging (OK) method. The analysis of Mean Square Error (MSE), Mean Absolute Error (MAE), Mean Error (ME) and the coefficient of determination (R^2) as the indices of the model performance indicated that they may significantly improve the predictive ability and thereby reduce the inherent risk in ore reserve estimation.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

Page

LIST OF TABLES

Page

Page

LIST OF APPENDICES

ACKNOWLEDGEMENTS

This research project was funded by the Department of Interior's Mineral Management Service through the University of Alaska Fairbanks' Marine Mineral Technology Center. Dr. Sukumar Bandopadhyay, Department of Mining and Geological Engineering, University of Alaska Fairbanks served as the principal investigator of the project.

I would like to express my deep sense of gratitude and heart felt thanks to my supervisor Dr. Sukumar Bandopadhyay, for his invaluable guidance, advice, encouragement, constant support and whole hearted cooperation during the course of my studies at UAF.

I am also grateful and express my sincere thanks to my committee members Dr. Falk Huettmann, Dr. Debasmita Misra and Dr. Rajive Ganguli for their advice, suggestions, encouragement and feedback on my work.

I specially thank Dr. Biswajit Samanta who without any hesitation always extended his help and offered suggestions whenever and wherever required. It is my pleasure to express gratitude to the staff and faculty members of the department for their support throughout this period.

I also extend my thanks to all my friends specially Debasish, Niladri, Dinesh, Abhijit, Neil and Pratap for their pleasant company within and outside the department. It is impossible to name all the people who helped me directly or indirectly in the course of this work. Therefore, I humbly extend my words of gratitude to all the inmates of the department and to all others, who might think they had been helpful to me in some way or another.

Last but not the least, I sincerely thank my parents, my wife and almighty God for their love, help and support all my life.

CHAPTER I

INTRODUCTION

1.1 Ore Reserve Estimation

The ore reserve estimation problem, essentially a statistical problem, can be stated simply as the determination of the value (or quantity) of the ore in unsampled areas from a set of sample data (usually drill hole samples) X_1, X_2, X_3, ...X_n collected at specific locations within a deposit. During this process it is assumed that the samples used for the inference of the unknown population or the underlying function responsible for the data are random and independent of each other.

The ore reserve estimation is usually a continuous process that begins during the exploration phase of a project and in some cases, continues throughout the life of the mine. At the early stage when sampling is conducted in widely spaced drill hole intervals, the estimates are basically global and have low confidence. In spite of the low confidence this is the first step at which mineral appraisal is carried out, and the objective of this estimation is to obtain a reasonable approximation of the grade-tonnage curve within a deposit confined by recognizable geological boundaries or a mineralized envelope. It also clarifies if further drilling is required. In that situation secondary drill holes are drilled at closer spacing to improve reliability. During the planning phase an estimate of the total recoverable reserves is made for various (i) cut-off grades and (ii) mining unit sizes. In this stage, the grade-tonnage curve is generated for blocks or for mining unit sizes. Based on the quality, quantity and location information of the ore grade obtained during this stage the subsequent mine operations are planned. Whatever the goal of reserve estimation, a reliable prediction is prerequisite for successful compilation of a mining project. Since the accuracy of grade estimation is one of the key factors for effective mining planning, design, and grade control, the estimation methodologies have undergone a great deal of improvement, keeping pace with the advancement of technology. There are a number of methodologies (Dutta et al., 2003; Dutta et al., 2006a; Dutta et al., 2006b; Samanta et al., 2005a, Samanta et al., 2005b) that can be used for the ore reserve estimation. The most common and widely used methods are the traditional geostatistical estimation techniques of kriging. Typically, the aforesaid criteria of randomness and independence among the samples are rarely observed. The samples are correlated spatially and it is this spatial relationship that is incorporated in the traditional geostatistical estimation procedure. This information is contained in a tool known as the "variogram function" which describes the continuity of the mineralization within a deposit both graphically and numerically. It can be also used to study the anisotropies, zones of influence and the variability of the ore grade values in the deposit.

Prior to the application of geostatistics, the ore reserve estimation methods were mostly empirical in nature. They consisted of the block methods (triangular, polygonal, and irregular) and the methods of cross section (vertical, horizontal, inclined). Recent advances in computational fields brought about the methods such as inverse distance weighing (IDW), which weighs the samples inversely with the distance from the point under consideration and combines them linearly.

Apart from the IDW methods, there are a number of kriging variants which are linear estimators. The most common is the ordinary kriging (OK) method, also known as the best linear unbiased estimator (BLUE). Unlike the other linear estimators the distinguishing feature of the OK method lies in its ability to produce estimates with minimum error variance. Even this best linear estimator of OK, may not however, perform satisfactorily under conditions of non-linearity. Under such situations, when non-linearity is present in the data, which is common in the complex phenomenon of ore reserve estimation, efforts should be made towards the use of non-linear estimators to improve the confidence in the ore reserve estimates.

The advent of modern computers has brought into light several machine learning algorithms that work in a quasi non-linear fashion. These artificial learning algorithms learn the underlying functional relationship inherently present in the data from the samples that are made available to them. The attractiveness of these non-linear estimators lies in their ability to work in a black box manner. Given sufficient data and appropriate training, they can learn the relationship between the input patterns (such as coordinates) and the output patterns (such as ore grades) in order to generalize and interpolate the ore grades for areas between drill holes. With this approach, no assumptions, such as linearity, are required to be made about any factors or relationships concerning the spatial variations of ore grade in the vicinity of boreholes.

1.2 Statement of the Ore Reserve Estimation Problem

The scope of the reserve estimation problem lies in the fact that an absolute or precise determination of the ore grade is not possible. It has always presented a challenge to mining engineers and geologists responsible for ore grade estimation. Reduction of the uncertainties in mineral appraisal invariably requires a reliable estimate of tonnage and grade of a deposit and grade control. Most of the ore deposits are formed under complex geological structures. The process of mineralization is largely affected by these geological structures which may include, among others, folds, faults, shear zones and joints. These are the potential sources of intrusion by other materials within the main deposits. Mineralization has led to the occurrence of ores in nature with widely varying properties. Generally, most ore deposits exhibit the following behaviors: (i) large variation of physical and chemical composition in both vertical and lateral extents, (ii) variations in deposition and evidence of structural disturbances, (iii) multiplicity of ore structures, (iv) variation in thickness and quality in the same structure, and (v) variation in the nature of the associated formation. As a result, estimation of ore grade and reserve are difficult for complex ore formations.

Since, an estimate of the ore grade includes a degree of uncertainty by the very nature of a deposit, it has led to the continual search for more reliable and robust estimation techniques. Apart from the nature of the mineralization and the complex geometry of an ore deposit, the choice of an estimation method is also dependent on the variability of the grade distribution, the characteristics of the ore boundary, the amount of resources available, extent of samples and the degree to which high grade outliers are present. There are numerous methodologies in use today which operate under fundamentally different concepts. Among the various techniques, the traditional approach has been the use of geostatistics.

In spite of the popularity of geostatistics in mineral appraisal, in recent times, researchers have opted and shown promising results in the field of predictive mapping using neural networks (Samanta et al., 2005a; Yama and Lineberry, 1999) and support vector machines (Kanevski et al., 2002; Pozdnoukhov, 2005). Since these artificial learning algorithms are trained from the samples that are made available to them, their efficiency of learning improves with an increase in the sample density. However, it must be realized that the sampling task in geological and mineral exploration is time consuming and expensive, and often the samples are noisy. Frequently, the sampling is done in wide drill-hole intervals, resulting in less representative data. Furthermore, the data are collected in non-optimal or near-optimal environments. As a result, the volume of data collected from drilling and sampling may be inadequate and even inappropriate to model a complex deposit. In such cases, due to the inherent sparseness and noise, ore reserve modeling becomes a challenging task. The reliability of an ore reserve estimate under such conditions is not only decreased but also has a low level of confidence. Since it is inherent in the process, efforts should be made to select an estimation method, which will treat the available data prudently and develop the necessary functional relationship needed for ore reserve estimation.

Selection of the estimation method aside, equally important in any modeling task is the validation of the model performance. One way to validate the model performance is to test the actual ore grade with the predicted ore grade. However, in the context of reserve estimation, it is almost impossible to compare the 'predicted' ore grade with the 'actual' ore grade. Several other factors during sample collection such as dilution, spillage, possible effects of stockpiling, possible sorting and concentration processes may hinder accurate comparisons. Therefore, in order to validate the model and its generalization ability several procedures such as bootstrapping, the split sampling method (or holdout method), the cross validation method (K-fold cross validation, leave-one-out cross validation) can be adopted. The basic idea of these techniques is to keep aside part of the data from the available dataset and not use them in the training process. These models, in general, learn the functional relationship from the training dataset. Thus, when the training is complete, the "partitioned" data will serve as the "new" dataset (known as the validation dataset) to assess the trained model performance. Each of these procedures has its own merits and demerits. The cross validation and split sampling method have been popular (Samanta et al., 2005b; Dutta et al., 2003; Twarakavi et al., 2006). Since, the predictive performance of the model depends to a large extent on the quality and the amount of data on which it is trained, the k-fold cross validation appears to be an appropriate choice when the dataset is sparse (Goutte, 1997). The disadvantage of this method is, however, that the training algorithm has to repeat *k* times, thus requiring additional computational time. Under such circumstances, the split sampling approach appears to be a better choice. It must be noted that with split sampling the results rely heavily on the distribution of the ore grade values in the training dataset and in the validation dataset. Since the learning models are built by exploring and capturing similar properties of the various data subsets, these data subsets should be statistically similar to each other and should reflect the statistical properties of the entire dataset. The statistical similarity ensures that the comparisons made for the model built on the training dataset and tested on the prediction dataset are logical (Bowden et al., 2002, Yu et al., 2003). Traditionally used practices of random division of data might fail to achieve the desired statistical properties when the data are sparse and heterogenious. Due to the sparseness, limited data points categorized into the data subsets by random division might result in dissimilarity of the data subsets (Ganguli and Bandopadhyay, 2003). As a consequence, overall model performance will be decreased. Therefore, careful subdivision of data during model development is essential. Various methodologies should be investigated for proper data subdivision under such a modeling framework.

1.3 Literature Review

Geostatistics has been the most used procedure for the complex phenomenon of ore reserve estimation (Journel and Huijbregts, 1978; Rendu, 1979; Pan, 1995). In recent times, several researchers have applied NN for ore reserve estimation. A representative application of NN for ore reserve estimation is reviewed below. Several artificial learning algorithms were applied for this purpose. Wu and Zhou, 1993 investigated a multi-layer feed forward neural network approach for copper reserve estimation. Initially, the network was trained with filed assay data at borehole locations and then was used to predict the distribution of ore grade in the drilling region. The NN results when compared with other traditional models indicated that after appropriate training on a comprehensive set of sample data, the NN could generalize reasonably well in the neighborhood of the sampling region.

Clarici et al. (1993) used the NN model for analyzing spatial data of drill hole locations, assay values (ppm) of arsenic, lead, and cadmium. The results when compared with kriging demonstrated the potential of NN as a tool for spatial data analysis. Denby and Burnett (1993) used GEMNET (grade estimation using mapping network) for estimation of grade in an iron ore deposit. Kapageridis and Denby (1998) presented a NN approach to model the ore grade spatial variability in a large undeveloped copper/gold deposit. They used a radial basis function network (RBFN) for the model development. The results indicated the potential of NN for ore reserve estimation. Others (Samanta et al., 2004a; Samanta et al., 2004b; Samanta et al., 2005a, Samanta et al., 2006) applied NN for the ore grade estimation of gold and bauxite deposits. The results in these studies indicated the potential of NN for the ore reserve estimation.

Dutta et al. (2006a) used a hybrid ensemble network model of NN and geostatistics to predict the ore grades in a bauxite deposit. Their study was based on the assumption that since kriging and NN capture different aspects of the spatial variability in the data, the hybrid model would give better estimates. Their study proved correct for the silica content of the bauxite. The alumina content was, however, predicted better with the kriging model. The failure of the hybrid model in the prediction of alumina was basically attributed to the high error of the individual NN models. For the same ore body, another study was also reported by using a Radial Basis Function (RBF) NN (Dutta et al., 2005d). In this study several important aspects related to RBF network modeling were discussed, including the appropriate division of the entire dataset into the modeling subsets using Genetic Algorithms (GA). Bowden et al. (2002), Samanta et al. (2004a), Samanta et al. (2004b), Ganguli and Bandopadhyay (2003) describe the importance of proper data division for development of model data subsets. They used different methodologies such as GA and Kohonen network for appropriate data division. Samanta et al. (2005a) used an ensemble NN model for the prediction of gold grades. The model consisting of multiple networks was constructed by applying the Adaboost algorithm using different training datasets. The purpose was to examine if the use of an ensemble model would provide better performance than the single neural network. There are several advantages of using an ensemble model. First, each neural network in the ensemble model follows more or less the true output mapping function. Conceptually, if one assumes that the output of an individual neural network of the ensemble consists of a true output plus a random error component with zero mean, then the combination of the outputs from the individual networks results in averaging of the random error components. Hence, it ensures reduction of the estimation error. Second, a single best network might get overfitted, thus, behaving poorly with unseen data. An ensemble of networks might reduce the overfitting, by combining different networks with different architectures. Third, the input-output relationship represented by a set of data with a distinct nature might not be captured adequately by a single network. It is possible to train individual networks using data having a nature to that of the entire set of data, and then combine the outputs of the individual networks to get a final improved ensemble output. Contrary to the information in the published literature (Sharky, 1999). Adaboost did not perform better than a single neural network in the cited application. The authors proposed a plausible reason to be the high noise inherent in the gold data used in the study. Multiple networks using the training data can also be constructed using a bagging or bootstrap aggregating technique (Breiman, 1996). In bagging, each network is independently trained on "n" samples picked randomly, with replacement from the "n" original samples of the training set. Each neural network is thereby trained on different but overlapping subsets of the original training data set, and will, therefore, give different predictions. Final prediction is the average of all the individual networks of the ensemble. When the Adaboost algorithm was used in an entirely different study (Dutta and Ganguli, 2005b) to determine the ash content of the raw coal in real time, the model performed appropriately. Dutta et al. (2003) also used an ensemble network for ore grade estimation. Their study revealed that the ensemble network performed slightly better than a single best neural network. Furthermore, in their application of an ensemble network for ore grade estimation, they selected the different networks by changing the network architectures and the number of hidden neurons, while the training data set was identical for each of the networks.

Application of NN has also been reported in several other mining applications such as in mineral processing plants (Hodouin et al., 1991), geological roof classification (Cardon and Hoogstraten, 1995), longwall stability prediction (Park et al., 1995), identification of failure models for underground openings (Lee and Sterling, 1992) and spatial continuity detection (Clarici et al., 1993). Apart from mining, it has also been applied to other related fields such as characterization of aquifer properties (Rizzo and

Doughetry, 1994), calibration of on-line analyzers (Yu et al., 2003) and ground water modeling (Rogers and Dowla, 1994), vegetation and land cover mapping (Fitzgerald and Lees, 1996; Foody, 1997), land degradation (Mann and Benwell, 1996), geological mapping (An et al., 1995), and classifying remote sensing data (Miller et al, 1995). Dutta et al. (2005c) used a multilayer feed forward NN and RBF neural network to predict the radioactivity levels at a given test site in Germany. In most of these studies it was not evident if these techniques provided a better estimated value than that of the geostatistical technique. In most of these studies it was revealed that neither the neural network nor the geostatistics proved superior to the other. The efficiency of the two techniques varied from one application to another.

Apart from NN, one more machine learning algorithm which is gaining popularity in the field of predictive mapping in several benchmark problems is the support vector machines (SVM). Although relatively new, this method is getting widespread acceptance because of its robust mathematical background (Kecman, 2000; Kecman, 2004; Smola and Scholkopf, 1998; Smola and Scholkopf, 2004). Also known as support vector regression (SVR), the method is based on statistical learning theory (SLT) and performs structural risk minimization (SRM). There are relatively few applications of SVM to mining reported in the published literature. This research is perhaps the first application of SVR for ore reserve estimation.

Mukherjee et al. (1997) have shown the remarkable predictive capability of the SVM algorithm. Their study revealed that SVM performs better than NN, RBF and local polynomial techniques when applied to a database of chaotic time series. Pozdnoukhov (2005) applied SVM to detect the natural radioactivity levels in a given test site. The data consisted of X-coordinate (m), Y-coordinate (m) and mean gamma dose rate (nanoSieverts/m). Analysis was performed on the two sets of data: one with the noise patterns and the other without any noise. The SVM method produced comparatively better results when compared to other techniques. Chang and Lin (2001) describe the various procedures ideal for the development of the SVM model. Cherkassy and Ma (2002) in their study investigate the various practical aspects in the selection of the SVM parameters in the SVM regression.

Kanevski et al. (2002) and Pozdnoukhov et al. (2002) also demonstrated SVM application to spatial data analysis in the presence of some priori knowledge. Twarakavi et al. (2006) applied SVM to predict the arsenic concentrations in the bedrock derived stream sediments using the gold concentration distribution present within the sediments. Their study was based on the hypothesis that arsenic displays a consistent correlation with gold, which is typical for gold deposits in general. Their study showed improved predictions compared with an earlier study in which NN was used for the same purpose (Misra et al., 2005). Twarakavi et al. (2006) also applied SVM to develop an optimal ground water quality monitoring network for a watershed. The water quality indicator considered in their study was the nitrate concentrations in the watershed. The long-term nitrate concentrations were modeled as a function of the land use distribution, recharge potential and the spatial co-ordinates. Though the developed model generated relatively large errors compared to other models, its lesser data requirements made it attractive. This is encouraging under the conditions of limited resources.

The general characteristic of SVM and NN emphasized the fact that they can approximate any multivariate non-linear relation among the variables in a black box manner and that both are robust to noisy data. The added advantage of the SVM algorithm lies in the fact that it not only tries to reduce the empirical error (the training data error) but also reduces the model complexity. The ability of the SVM to work with small datasets is extremely useful. SVM may be able to capture the spatial distribution of ore grade more effectively with careful modeling and selection of SVM parameters. Therefore, the purpose of the present study is another attempt to investigate the applicability of machine learning algorithms for ore reserve estimation.

1.4 Scope of the Study

A number of state-of-the-art models are available for predictive spatial mapping. The application of these tools has been made possible due to recent advances in the computational platforms. The application of these techniques include but are not limited to the method of geostatistics: the family of kriging estimators (Isaaks and Srivastava, 1989; Samanta et al., 2005a), machine learning algorithms such as Support Vector Machines (SVM); neural networks (NN), (Samanta et al., 2005b, Ganguli and Bandopadhyay, 2003; Dutta et al., 2005a, Yu et al., 2003) and hybrid models (Dutta et al., 2006a, Kanevski et al., 1996). The focus of this study is the application of the machine learning algorithms such as NN and SVM for ore reserve estimation. The working principle of SVM makes it robust against noisy and extreme value data. At the same time, it can capture the high-dimensional non-linear spatial trends if they exist in the data. This noise and complexity are predominant in the mining domain. While the family of kriging estimators is popularly used in various fields, their performance depends to a large extent on the presence of good and sufficient data to map the spatial correlation structure. They also work better if there is a linear relationship between the input and the output patterns. However, this is rarely the case in the mining domain. Even though there are a number of kriging variations, such as lognormal kriging and indicator kriging that apply certain specific transformations to capture the nonlinear relationships, they may not be sufficient to capture the broad nature of spatial nonlinearity. Moreover, earth sciences data are most often characterized by the presence of noisy patterns, that are also of unknown nature and are usually difficult to discernr. The SVM is effective for modeling using sparse datasets, because it only uses a few data points as features vectors for defining the model. Further, with SVM, there is no need to perform semi-variogram modeling, which is the core of the geostatistical estimation method. It works like a black box. With semi-variogram modeling, it is preferable to have the data normally distributed, which is usually not the case. This can be avoided while performing SVM modeling. Geostatistical techniques such as ordinary kriging work under the assumption of stationarity. However, in most deposits this might not be truly observable. Such an assumption is not required with SVM modeling. Also, at times, with geostatistical techniques, the anisotropies may not be evident in a particular direction when the samples are sparse. This may lead to unreliable estimates. Such a situation can be avoided while employing the SVM modeling. It has its own advantages when compared with NN. Although NN models are also a powerful tool to capture the nonlinear spatial relationships that may be present in the data, they are usually difficult to optimize under sparse data settings. Of the various NN alternatives, multilayer feed forward networks (MFFN) have been successfully applied in several fields (Samanta et al., 2005b; Dutta et al., 2005c). Despite their effectiveness, the model selection and estimation process is typically difficult, time consuming and computationally intensive, as it involves solving complex integration and/or optimization of parameters. Furthermore, they are susceptible to local minima and in the presence of a large number of local minima, the NN may fail to estimate the global minima. In SVM modeling, however, estimating the unknown parameters only involves optimization of a convex cost function. This can be achieved using standard quadratic programming algorithms (Kecman, 2004). The model constructed depends explicitly on the most "informative" data (the support vectors). From the previous sections it can be very well perceived that the extent to which various methodologies affect the grade estimation is quite variable. Therefore, the scope of this research includes the following objectives:

1) To examine the effect of the various data divisional approaches on the model performance, since the model datasets have a significant impact on the model generalization ability.

2) To develop a reserve model using machine learning algorithms (the support vector machine approach and the neural network approach) for improved ore grade estimation. Although successfully implemented in other fields (Pozdnoukhov, 2005; Dutta et al., 2005c; Kanevski et al., 2002), there is no known application of SVM to the ore reserve estimation problem. Two case studies have been carried out utilizing the actual drill-hole information. The first dataset is a placer gold drillhole data. The data are

very noisy and sparse. The drilling of holes is not on a regular grid and the hole spacing is often too large to apply geostatistics in order to calculate placer gold reserve accurately. The second dataset is a lode deposit and is continuous in nature.

3) To apply the SVM and NN model on the placer gold dataset and compare the grade estimates with the traditional ordinary kriging method and develop the volume of the reserves for various cut off grades.

4) To develop alternative mining blocks using the placer gold data, by clustering algorithms, and calculate the volume of reserves for various cut off grades.

CHAPTER II

THEORY OF GENETIC ALGORITHMS AND KOHONEN NETWORK

2.1 Genetic Algorithms and Data Division

Genetic algorithms (GA) are a search procedure based on the mechanics of genetics and natural selection. The advantages of GA in data divisional problem are that GA generates optimal data divisions quickly after examining only a small fraction of the search space in data divisional space. Genetic algorithms combine an artificial survival of the fittest approach with the various genetic operators to form a mechanism from which optimal solutions may eventually be produced for data division.

In nature, organisms evolve as the result of selective processes, such as mating between individuals, and occasional mutations. Genetic algorithms mimic these same operations and employ several operators that duplicate, recombine, and change the string of a current solution to create a new solution. These operators are known, respectively, as reproduction, crossover and mutation. Reproduction and crossover play the primary roles in an artificial genetic search. Reproduction emphasizes highly fit strings while crossover recombines these selected solutions to generate new, potentially better solutions. Mutation plays a secondary role in producing optimal solution by introducing the occasional original change in a solution. Mutation provides a mechanism to escape from a false local optimal solution through occasional alteration of the solution. Thus, genetic algorithms are recognized as global learning algorithms. The principle stages of genetic algorithms are shown in Figure 2.1 (Dutta et al., 2006a).

The genetic optimization of a data division is carried out in a manner similar to that described above. A data division is performed by selecting members of the sample in such a way that the first 50% of the selected samples are put in training set, the next 25%

are placed in a calibration set and the remaining 25% into a validation set. In an optimal data division, samples are ordered in such a way that statistical differences between the three subsets are minimized. The methods and procedure for GA data divisions are described in the following sections and illustrated through a simple example in Figure 2.2. The following steps are used for generating data divisions using genetic algorithms:

(a) Generation of random solutions for data division

Random solutions are created by arbitrarily ordering the samples, and splitting the dataset such that the first half is put into the training subset, the next quarter into the calibration subset and the remaining (quarter) into the testing subset. To start the process, a suite ("population") of solutions is generated. For example, assume one has eight samples to divide; division should occur so that the first four selected samples fall in the training set, the second two samples fall into the calibration set, and the remaining two samples are placed into a validation set. Random data divisions could be generated in the way shown in Figure 2.2. In this figure it is shown that a population of 20 random solutions could be created by different orderings of the samples. Note that numbers in the cells indicate the sample number (sample I_D) and the position of the cells indicate the sample order.

(b) Assessment of the fitness values

The next step involves assessing the quality of the generated solutions. The quality of a solution is determined by its "fitness" value. Fitness value is the criterion upon which a solution can be judged. In the present data divisional application, a criterion was developed which minimized the mean squared deviation as well as variance among the three subsets and the entire data set.

Note that the *only* intent of using GA is to ensure that the three subsets are statistically equal, i.e. the constituents of each subset are statistically similar to the corresponding constituents of the other two subsets. In the GA algorithms all the variables are taken into consideration.

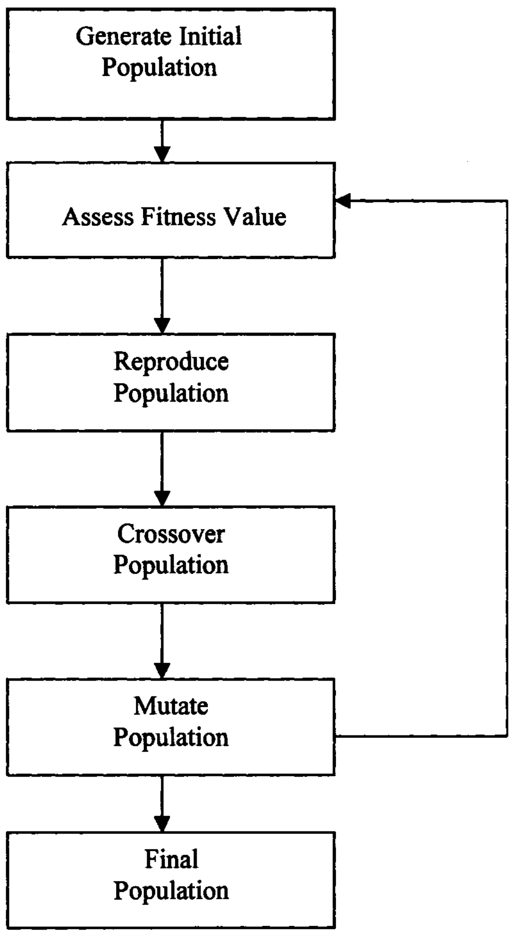


Figure 2.1: Principal stages of genetic algorithms

(c) Reproduction of the solutions

The population of the solutions is then modified to obtain the next "generation" of solutions. This is done in two steps: first by selecting survivors and second by "evolution" of the survivors. Selection of survivors is done by remainder stochastic sampling (Golderg, 1989) with the constraint that solutions with better fitness values have a higher chance of survival. A particular solution may be selected many times, while another may not be selected at all (Figure 2.2). At this point, the population consists of the "good" quality solutions from the previous populations, sometimes with multiple copies of the same solutions. As a result, average fitness of the solutions in the population is increased.

(d) Crossover of the solutions

During the crossover operation, solutions are randomly combined in pairs on a probabilistic basis (Figure 2.2). The individual solutions in a given pair ("parent") are then modified by the crossing over of features between the solutions. It is important to remember that selected pairs and their respective cross-over points are chosen randomly. Crossover results in the crossed pair having modified characteristics of the parent solutions. Some of the modified solutions have superior fitness values, improving their chances of survival into future generations, whereas some have inferior fitness values, reducing their chances of survival. Normally, crossover involves swapping blocks of samples. For example, in Figure 2 the crossover point randomly generated for the parent-pair is 4 blocks down in a string of 8 samples. Crossing over then results in a single solution containing samples in slots one through three of parent #1, and samples in slots four through eight of parent #2, while the other solution consists of samples in slots one through three of parent #2, and samples in slots 4 through 8 of parent #1.

(e) Transformation into feasible solutions

Crossover results in infeasible data divisions in the sense that some duplicate samples are found in the solution and some samples are left off. For example, after the crossover, solution #2 (Figure 2.2) contained the duplicate samples three and eight. On the other hand, samples # 1 and 4 are not selected in any of the slots. Therefore, it is necessary to replace duplicate samples with the left off samples, aiming to do so with minimal disturbance of the solutions.

(e) Mutation of the solutions

Mutation is performed on a probabilistic basis, where a sample from one subset is randomly swapped with a member of another subset. For example, a solution may be mutated by randomly swapping samples (3 and 87). The mutation of the solutions helps to maintain genetic diversity and prevents the system from converging to a false optimum solution.

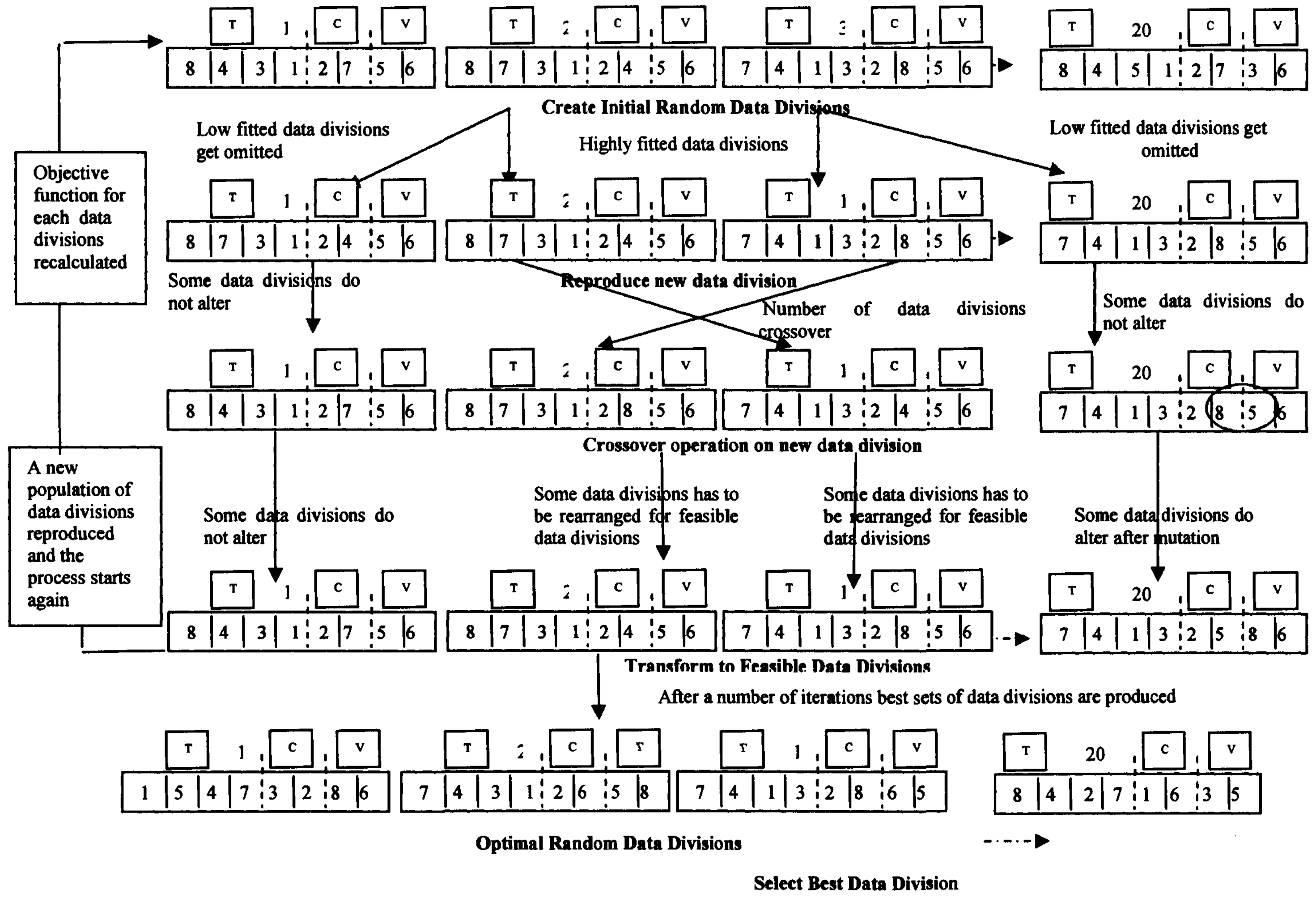

Create Initial Random Data Divisions
Low fitted data divisions get omitted
Highly fitted data divisions
Low fitted data divisions get omitted
Objective function for each data divisions recalculated
Some data divisions do not alter
Reproduce new data division
Number of data divisions crossover
Some data divisions do not alter
Crossover operation on new data division
A new population of data divisions reproduced and the process starts again
Some data divisions do not alter
Some data divisions has to be rearranged for feasible data divisions
Some data divisions has to be rearranged for feasible data divisions
Some data divisions do alter after mutation
Transform to Feasible Data Divisions
After a number of iterations best sets of data divisions are produced
Optimal Random Data Divisions
Select Best Data Division

Figure 2.2: Genetic Algorithms for Data Divisions

2.2 Self Organizing Map and Data Division

The Self-Organizing Map (SOM) was introduced by Teuvo Kohonen. The SOM (also known as the Kohonen feature map) algorithm is one of the best known artificial neural network algorithms. In contrast to many other neural networks, which require using supervised learning, the SOM allows unsupervised learning. The SOM algorithm employs a technique known as competitive learning. All neurons in the output layer compete with each other in response to a particular input pattern, with only a single output neuron winning the competition to become activated. The winning neuron upon activation excites neighboring neurons, changing their respective weights in the learning process. In SOM, the output neurons are located on a one or two-dimensional lattice. The neurons in the lattice are selectively tuned to various input patterns during training. As a result, the locations of the neurons in the lattice become ordered in such a way that a coordinate system for different input patterns is created over the lattice. The basic idea of SOM is to define a one or two-dimensional map of output neurons from a higher dimensional input space. Each output neuron carries a reference location of a particular input pattern or group of similar patterns in the lattice. The output neurons are ordered in such a way that their neighborhood relation is dictated by the topological maps. This means two neurons have more in common if they are located adjacent to each other than if they are some distance apart. Thus the SOM algorithm provides a non-linear clustering mechanism in which similar patterns can be grouped into an output neuron in the lattice.

Learning of a SOM network involves essentially three major tasks: competition, cooperation, and synaptic weight adaptation. For each input pattern, neurons in the output layer compute their respective distances to the input pattern. The neuron with the minimum distance wins the competition. A topological neighborhood is then defined around the winning neuron, in which neurons cooperate amongst themselves. The winning neuron locates itself in the center of the topological neighborhood of excited neurons. The excited neurons in the neighborhood respond by updating their synaptic weights in response to the input pattern, so that they are attracted towards any similar input patterns.

To start the learning process, an algorithm proceeds by initializing the synaptic weights in the network. This is generally done by assigning small values from a random number generator. The basic stages of the SOM learning process is as follows (Lippmann, 1987):

1. One sample vector x is randomly drawn from the input data set and its similarity (distance) to the output neurons is computed, e.g. by using the common Euclidean distance measure:

$$\|x_i - w_c\| = \min_i(\|x - w_i\|) \tag{2.1}$$

The best matching neural of the output layer is found. W_i=weight vector of neural i in the output layer

2. After the best matching unit has been found, the synaptic weights of the output neurons are updated. The best matching unit and its topological neighbors are moved closer to the input vector in the input space, i.e. the input vector attracts them. The magnitude of the attraction is governed by the learning rate. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to the specified learning rate function type. Along with the learning rate, the neighborhood radius also decreases as time progresses. The update rule for the reference vector of unit i is the following:

$$w_i(t+1) = \left\{ \begin{matrix} w_i(t) + \alpha(t)(x(t) - w_i(t)) \\ w_i(t) \end{matrix} \right\} \begin{matrix} i \in N_c(t) \\ i \notin N_c(t) \end{matrix} \tag{2.2}$$

Where, $\alpha(t)$ is a scalar value adaptation with gain $0 < \alpha(t) < 1$, which depends upon learning rate and neighborhood distance, and where N_c is the search neighborhood

3. The steps 1 and 2 together constitute a single training step and they are repeated until the training ends. The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated off of this value. After the training is over, the map should be topologically ordered.

The intent of SOM for this study is to use it as a clustering technique to group similar patterns. Since SOM assimilates similar patterns into a group, random selection of the samples from each group for training, calibration and validation will result in each subset of data acquiring all the larger diversified patterns. Therefore, three subsets of data will be fed, with similar types of patterns. To cluster the data, all the inputs and output are presented to the network as SOM's input. The output of SOM is obtained in terms of a number of output groups, and each group contains the winning patterns.

CHAPTER III

THEORY OF SUPPORT VECTOR MACHINES

3.1 Basics of Supervised Learning from Data

The process of supervised learning from the data involves the use of three basic components. They are the random input variable **x**, the system response variable y and a learning machine that determines the unknown dependency between the higher dimensional input vector **x** and the system responses y. During the learning phase the learning machine detects the relationship between the input and the output variable from the available data D in the regression task (or finds a decision boundary that separates the data for the classification tasks). The result of a successful learning process is an "approximating function" f_a (**x**,w) which in the statistical literature is also known as the hypothesis function. This function belongs to a hypothesis space of function H ($f_a \in H$) and approximates the underlying (or true) dependency between the input and the output in case of regression and the decision boundary in case of classification. It also tries to minimize the associated risk function R (**w**). Such type of learning is also known as distribution free learning because there is no information available about the underlying joint probability distribution.

The approximating function described in the preceding paragraph is a mathematical structure that can map the inputs x into the output y. It can be a multilayer perceptron, a RBF network, fuzzy model or various other mathematical models. But in this chapter, a limited discussion of the support vector machines (SVM) is given. A detailed discussion on this subject can be found elsewhere (Kecman, 2002; Kecman, 2004). Unlike the classic statistical inference problems, development of SVM is generally appropriate for the following contemporary problems:

1) Modern problems are high dimensional. If they were to be solved based on the linear assumptions of the contemporary techniques it would have resulted in an exponential increase of the number of terms known as the "curse of dimensionality."

2) In most situations, the underlying data generation laws may be far from normal.

3) In most practical situations collection of data is an extremely difficult task. In such sparse settings modeling can be difficult.

4) Because of the first two problems, the maximum likelihood estimator, and consequently the sum of error square assumption on which the classical techniques are based, are replaced in SVM by a new induction paradigm called the structural risk minimization (SRM) to model the non-Gaussian distributions.

In order to develop a model with a good generalization property two basic constructive approaches (Vapnik, 1995; Vapnik, 1998) must be used:

1. Selection of an appropriate model structure (number of hidden layer, number of hidden layer neurons, order of polynomial, number of rules in the fuzzy logic model) with the estimation error (a.k.a. variance of the model) fixed, and minimizing the training error (i.e. empirical risk).
2. For the training error (a.k.a. approximation error, empirical risk) fixed (equal to zero or some acceptable level) minimization of the estimation error.

Classical NN implement the first strategy while SVM implement the second approach. The goal of both the approaches is to match the learning machines capacity, to the training data complexity. The only difference between them is the approach taken for the minimization of different cost functions. Table 1 tabulates the basic risk functions applied in developing the three statistical models.

Table 3.1 Basic models and their error functions

Multilayer Perceptron (Neural Network)	Regularization Network (RBF network)	Support Vector Machine
$R=\sum_{i=1}^{l}(d_i-f(x_i,w))^2$ closeness to data	$R=\sum_{i=1}^{l}(d_i-f(x_i,w))^2+\lambda\,\|Pf\|^2$ closeness to data smoothness	$R=\sum_{i=1}^{l}L_g+\Omega(l,h)$ closeness model capacity

In Table 3.1, d_i is the desired value, w is the weight vector, λ is the regularization parameter, P is the smoothness operator, L_g is the SVM loss function, h is the Vapnik Chervonenkis (VC) dimension and Ω is known as the confidence term bounding the capacity of the learning machine. It could be seen from the table that unlike the classic algorithms such as NN and RBF, the SVM represents a novel learning technique which performs SRM. In general, the working of the SVM ensures that it creates a learning model with a minimized VC dimension. When the VC dimension of the model is low the expected probability of error is low as well. This in turn indicates good performance on the previously unseen data, i.e., a good generalization performance. The following sections briefly describe the theory and the methodology involved in the working of the regression aspect of support vector machines. A detailed description can be found elsewhere (Kecman, 2000; Kecman, 2004).

3.2 Support Vector Machines in Regression

The support vector machines comprise a set of powerful tools to perform classification and regression tasks. Apart from being systematic and principled, this approach, motivated by Statistical Learning Theory (SLT), has become very popular recently in the machine learning community. The regression aspect of SVM, known as the support vector regression (SVR) is based on the structural risk minimization (SRM) principle. While performing the regression, the SVR acquires knowledge from the experimental data in order to generalize to the previously unseen data. In general, they are non-parametric models based on supervised learning techniques because the parameters that define the capacity of the model are not predefined, rather their values are data driven in such a way as to match the model capacity to the data complexity. With spatial data this involves prediction of unknown mapping between the input (spatial co-ordinates, secondary variables) and the output (variables of interest) variables.

3.2.1 Statistical Learning Theory (SLT):

The process of SLT involves learning from relatively few training data points during which the expected risk R is approximated and minimized by the empirical risk R_{emp}. This is the induction principle of empirical risk minimization (ERM). However, finding the minimum of empirical risk is an 'ill-posed' problem due to the infinite number of possible solutions or approximating functions that are available when the learning machine is trained using a particular subset of the true underlying function. The approximating functions are always biased depending on the specific training data pairs. The process of minimization of the expected risk by developing a model from the training data will always include a generalization error. The generalization error bound is given by

$$R(h) \leq R_{emp}(h) + \Omega(h) \tag{3.1}$$

Where, R is the bound on the testing error, R_{emp} is the empirical risk on the training data and Ω is the confidence term which depends on the complexity of the modeling function. The parameter that defines the model complexity is termed as the VC dimension (h). It is introduced in the SLT to describe the general notion of "complexity", and is independent of any particular function used to model the data. Figure 3.1 shows the variation of the confidence term Ω w.r.t VC dimension (h) while Figure 3.2 shows the variation of the error w.r.t. the VC dimension (h).

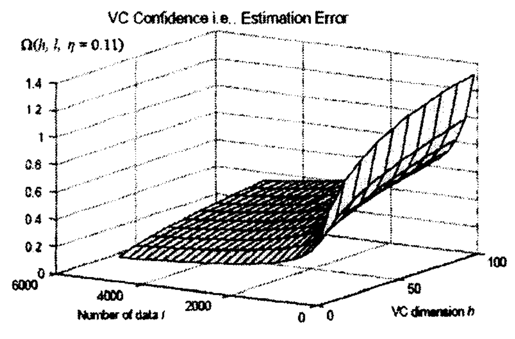


Figure 3.1: Dependence of VC confidence Ω on the VC dimension h and training data l, h<l
(from Kecman, 2000)

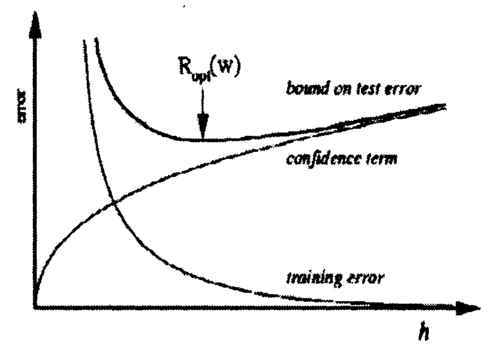


Figure 3.2: Bound on the test error derived in SLT. The minimum corresponds to an optimal model complexity
(from Pozdnoukhov, 2005)

According to the bounds derived in SLT, the general strategy is to learn from finite training datasets and choose, from all the possible candidates, an appropriate model that minimizes the training error and has the smallest VC dimension (Figure 3.2). This is the principle of SRM, which in turn results in the smallest bound on the test error. While SRM results in improved generalization of the learning machine, algorithmically it is realized through SVM.

3.2.2 Support Vector Regression (SVR)

In SVR, given a training dataset {(x,y), i=1,....L} where x_i are the inputs and y_i are the measured values, a functional dependence between the two variables is established. The idea is to minimize the empirical risk by introducing a novel loss function termed as the Vapnik's linear loss function with ε- insensitivity zone. (Equation 3.2)

$$\begin{aligned} E(x,y,f) = |y-f(\mathbf{x},\mathbf{w})|_{\varepsilon} &= 0 && \text{if error} < \varepsilon \\ &= (|y-f(\mathbf{x},\mathbf{w})| - \varepsilon) && \text{if error} > \varepsilon \end{aligned} \quad (3.2)$$

The error less than the 'ε' is ignored. This error loss function defines an error tube as shown in Figure 3.3. For $\varepsilon = 0$, Vapnik's loss function equals a least modulus function and in that case the SVR performs an interpolation of the training data.

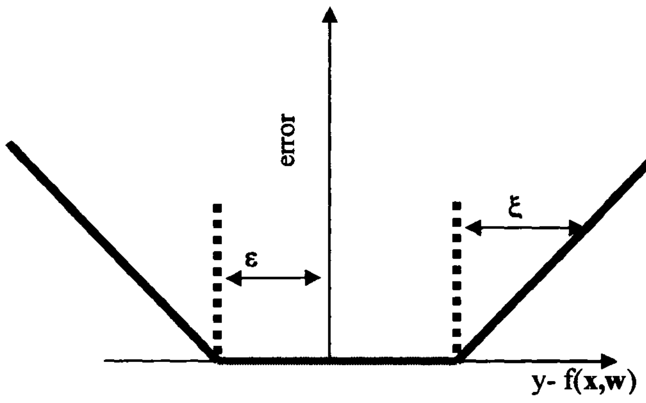


Figure 3.3: ε insensitivity loss function

The objective in SRM is to select an approximating function that not only minimizes the confidence term but also the empirical risk. The confidence term *Ω* is minimized by the minimization of $||w^T w||$ term. The overall risk that is minimized is given by the following objective function:

$$R = ||w^T w|| + C\sum| y - f(\mathbf{x},\mathbf{w})|_{\varepsilon} \tag{3.3}$$

Where, C is the penalty parameter known as the cost function.

For training data outside the error tube,

$|y-f(\mathbf{x},\mathbf{w})|_{\varepsilon} = \xi$, for data "above" the error tube, i.e. positive errors.
$|y-f(\mathbf{x},\mathbf{w})|_{\varepsilon} = \xi^*$, for data "below" the error tube, i.e. negative errors.

Thus, the equation (3.3) can now be written as

$$\text{minimize} \quad R = ||w^T w|| + C[\sum\xi_i + \sum\xi_i^*] \tag{3.4}$$

Under the constraints,

$$y_i - w^T x - b \leq \varepsilon + \xi_i \tag{3.4a}$$

$$w^Tx+b-y_i \le \varepsilon + \xi_i^* \tag{3.4b}$$

$$\xi^*_i \ge 0,\ \xi_i \ge 0 \tag{3.4c}$$

where, ξ_i, ξ_i^* are the positive slack variables shown in Figure 3.3 for measurement above and below the tube. Since this is a classic optimization problem with inequality constraints, it can be solved by using lagrange multipliers in a systematic manner employing the Kharush-Kuhn-Tucker (KKT) method. During this a primal Lagrange function L_p is created by subtracting from the objective function the constraints multiplied by corresponding Lagrange multipliers α. The KKT method suggests the conversion of an inequality constraint of the form $h(x) \ge (\text{or} \le) = 0$ into an equation of the form $h(x) = 0$ by adding or subtracting (depending upon the optimization problem) slack variables and then solving the corresponding equality constrained quadratic optimization problem (Miller, 2000). The risk function (in terms of the corresponding independent primary variables) in equation 3.4 can be expressed as a primal Lagrange function by

$$L_p(w,b,\xi_I,\xi_i^*,\alpha,\alpha^*,\beta,\beta^*) = (½)\|w^Tw\| + C[\sum\xi_i + \sum\xi_i^*] - \sum(\beta_i^*\xi_i^* + \beta_i\xi_i) - \sum\alpha_i^*(y_i - w^Tx - b + \varepsilon + \xi_i) - \sum\alpha_i(w^Tx + b - y_i + \varepsilon + \xi_i) \tag{3.5}$$

At the optimal point the first partial derivative of the Lagrange function w.r.t. the independent variables ($w, b, \xi_I, \xi_i^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*$) vanishes.

$$dL_p/\,dw = w_0 - \sum(\alpha - \alpha^*)x_i = 0 \tag{3.5a}$$

$$dL_p/\,db = \sum(\alpha - \alpha^*) = 0 \tag{3.5b}$$

$$dL_p/\,d\xi_i = C - \alpha_i - \beta_i \tag{3.5c}$$

$$dL_p/\,d\xi_i^* = C - \alpha_i^* - \beta_i^* \tag{3.5d}$$

Also, the following complementary conditions must be satisfied for the first partial derivative of L_p w.r.t. Lagrange multipliers ($\alpha_i, \alpha_i^*, \beta_i, \beta_i^*$) and slack variable (s).

$$\alpha_i(w^Tx + b - y_i + \varepsilon + \xi_i) = 0, \tag{3.5e}$$

$$\alpha_i^* (y_i - w^T x - b + \varepsilon + \xi_i) = 0 \quad (3.5f)$$

$$\beta_i^* \xi_i^* = (C - \alpha^*)\xi_i^* = 0 \quad (3.5g)$$

$$\beta_i \xi_i = (C - \alpha)\xi_i = 0 \quad (3.5h)$$

when the equations 3.5a-3.5b are substituted into the primal L_p in equation 3.5, it becomes a dual variable Lagrangian $L_d (\alpha_i, \alpha_i^*)$ given by

$$\max \quad L_d (\alpha_i, \alpha_i^*) = (-1/2) \sum(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i^T x_i - \varepsilon \sum(\alpha_i - \alpha_i^*) + \sum(\alpha_i - \alpha_i^*)y_i \quad (3.6)$$

subjected to

$$\sum(\alpha_i - \alpha_i^*) = 0 \quad (3.6a)$$

$$0 \leq \alpha_i \leq C \quad (3.6b)$$

$$0 \leq \alpha_i^* \leq C \quad (3.6c)$$

The solution of this standard quadratic optimization problem results in ‘l’ pairs of (α_i, α_i^*), one for each of the training patterns. The pairs that result in non-zero α_i or α_i^* are termed as the support vectors and they are the ones which influence the model. The complexity of the model is directly proportional to the number of support vectors. In this way the best regression hyper plane is given by

$$\begin{aligned} f(x,w) &= w_o^T x + b \\ &= \sum(\alpha - \alpha^*) x_i^T x + b \end{aligned} \quad (3.7)$$

When doing the non-linear regression model, a mapping function Φ (x) {such as polynomial kernels, Gaussian kernels} will be used to map the input space into the higher dimensional feature space and then construct the linear regression hyper plane in the feature space. The basic working principle remains the same.

The process of making a prediction employing the SVR can be shown graphically (Figure 3.4). It presents an overview of the different steps involved in the support vector regression to make a prediction. They can be summarized as follows:

1) Map the test pattern **x** into the feature space by a mapping function φ. The mapping function is selected during the model development stage.

2) Compute the dot product with the images of the training pattern mapped on φ. This is equivalent to evaluating the corresponding kernel function k (x_i, x).

3) Add the dot products using the weights $v_i = \alpha_i - \alpha_i^*$.

4) Add the constant term b to derive the final model output for the pattern **x**.

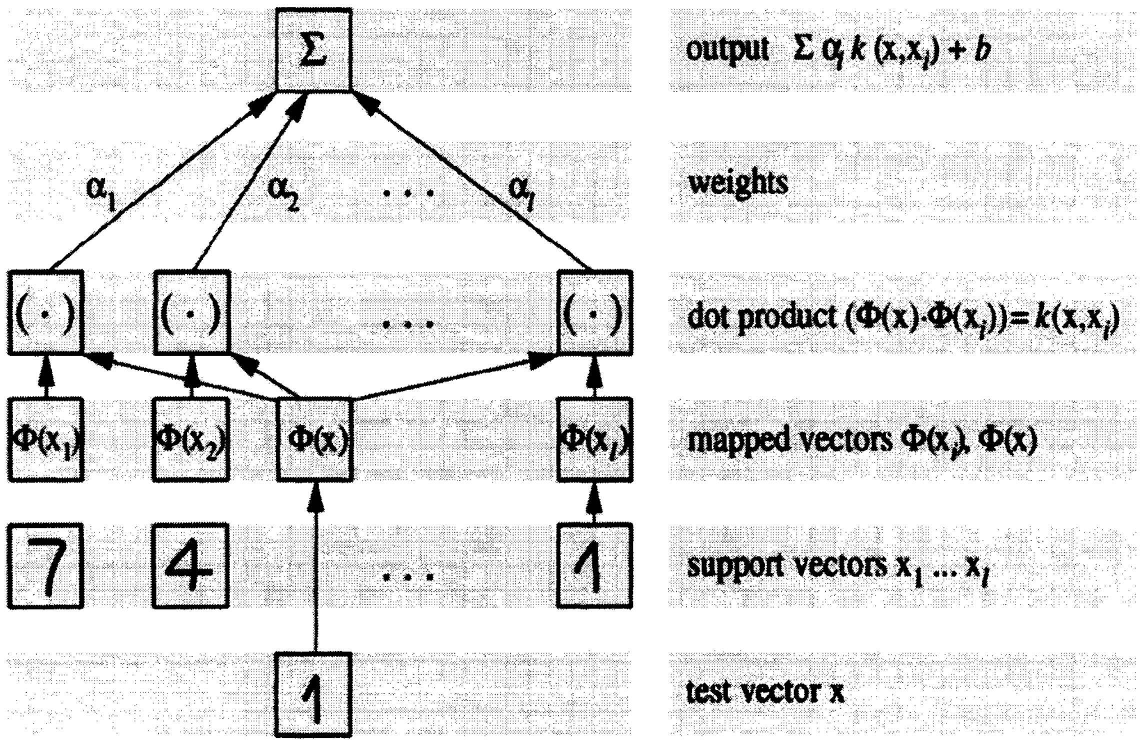

Σ
output $\Sigma \alpha_i k(x,x_i) + b$
α_1
α_2
...
α_l
weights
(·)
(·)
...
(·)
dot product $(\Phi(x)\cdot\Phi(x_i)) = k(x,x_i)$
$\Phi(x_1)$
$\Phi(x_2)$
$\Phi(x)$
$\Phi(x_l)$
mapped vectors $\Phi(x_i)$, $\Phi(x)$
7
4
...
1
support vectors $x_1 \dots x_l$
1
test vector x

Figure 3.4: Architecture of a regression machine constructed by the SVM (from Smola and Scholkopf, 1998).

3.3 SVM Model Development

The training and design of the SVM model is an iterative process and essentially it consists of performing the following basic steps (Chang and Lin, 2001; Kecman 2004):

1) Define the problem as classification or regression.

2) Standardize the input data.

3) Check for outliers, i.e. the strange data points.

4) Select the kernel function in order to transform the data to a higher dimensional feature space. This helps determine the hypothesis space of the decision or regression function in the classification or regression problem. One of the common kernels considered is the RBF kernel.

5) Select the 'shape', i.e., the smoothing parameter σ of the kernel function. This is the polynomial degree for the polynomial and variances for the Gaussian RBF kernel.

6) Choose the penalty parameter C and the desired accuracy defining the insensitivity zone ϵ.

7) Solve the quadratic programming problem in the 2 x L variables for the corresponding regression task.

8) Train the model and validate it on a previously unseen dataset. If the validation result is not satisfactory, repeat the steps from 4 to 8.

9) Since the search for the individual C, ϵ and the shape parameter σ can be tedious and a time consuming task, an alternative approach could be cross-validation and grid search to find the best value of the cost parameter.

CHAPTER IV

ADDRESSING THE SPARSE DATA ISSUE IN NOME

4.1 Placer Gold in Nome

The Nome district is located on the south shore of the Seward Peninsula at roughly latitude 64°30' north and longitude 165°30' west. It is 840 km west of Fairbanks and 860 km northwest of Anchorage (Figure 4.1). The first discovery of placer gold in Nome dates back to 1898. Gold and antimony have been produced from lode deposits in this district and tungsten concentrates have also been produced from residual material above the scheelite-bearing lodes near Nome. Other valuable metals, including iron, copper, bismuth, molybdenum, lead, and zinc, are also reported in the Nome district.

Hopkins and MacNeil (1960) among others, studied these deposits, recognizing their origins and chronicling their exploration, and speculated on their chronology and the events in the complex regional glacial history that allowed their formation and preservation. Figure 4.2 shows the composition of the offshore placer gold deposit. Due to the extent and richness of the Nome gold resources, the area was studied extensively, and geological, geophysical, and geo-chemical characteristics of near-shore gold deposits are well documented in the published literature. Rusanowski (1994) presented an excellent summary of the Nome Offshore placer project, and some of the relevant details on the Nome Offshore placer project from Rusanowski's work are reproduced here as a foundation for the present research.

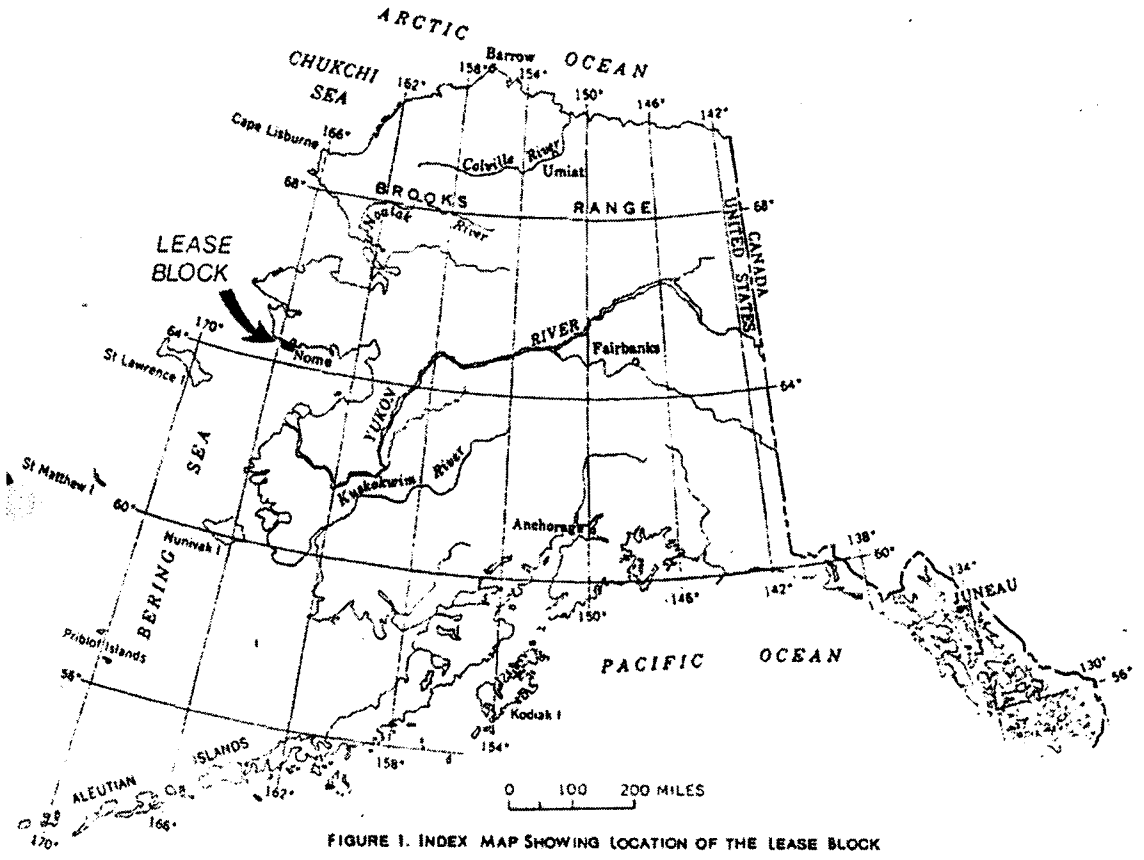


Figure 4.1: Location of the Nome area (from Rusanowski, 1994)

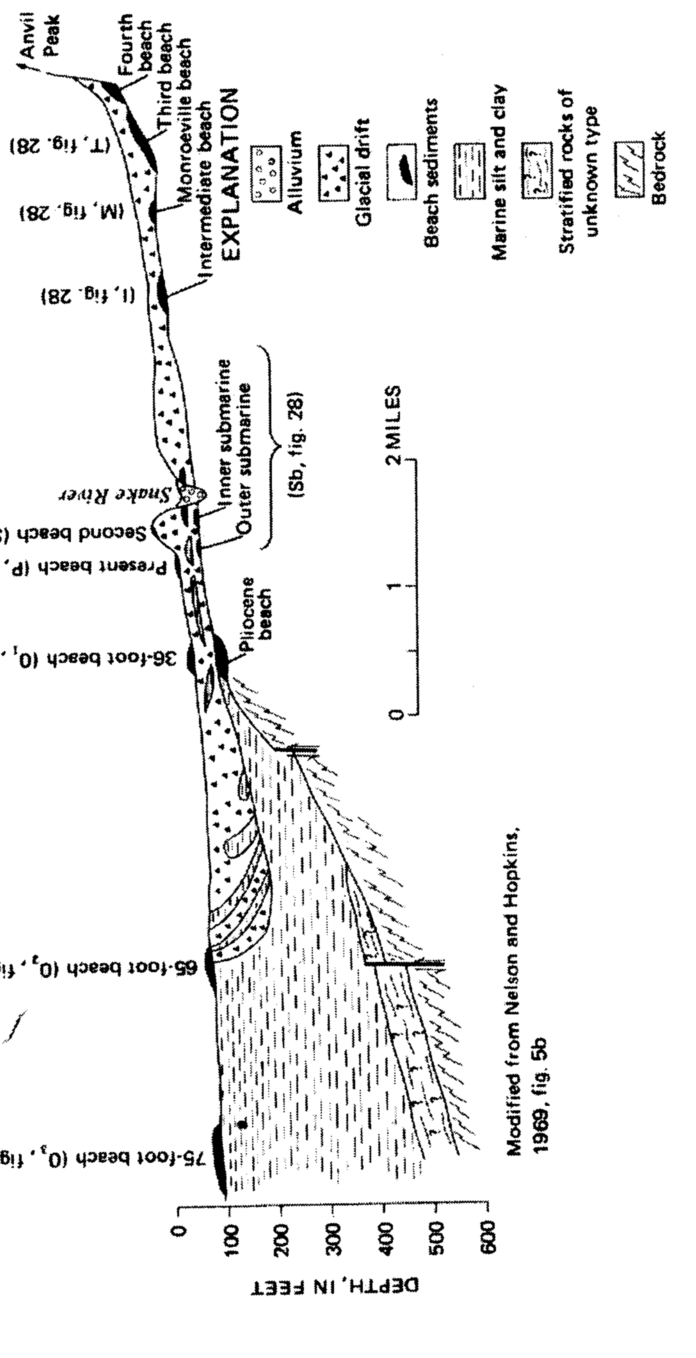


Figure 4.2: Offshore placer gold deposit (from Cobb, 1974)

4.1.1 The History of Offshore Exploration

There is a long history of offshore gold exploration off the coast of Nome in Alaska. Shell Oil Company conducted a seismic and magnetic survey of the area in the year 1962. After an extensive study in 1963, a drilling program was conducted to sample an area of about 22,000 acres. The program resulted in the completion of 568 drill holes in 1964.

In 1967 and 1968, the U.S. Geological Survey collected 700 bottom samples in the northern Bering Sea and sampled the beaches at Nome and other areas around Norton Sound. In addition, the U.S. Bureau of Mines drilled 51 holes offshore around Nome.

Asarco, Inc. took over the operation in 1968 and drilled an additional 500 holes through the ice on the newly optioned leases west of Nome in early 1969. Asarco also carried out investigations including bottom photography, current and wave measurements, bulk sampling, pilot scale mining, environmental assessment, and a preliminary engineering design and cost estimate. However, the economics were not favorable based on the price of gold at that time ($35 per ounce).

In 1983, Power Resources Corporation contacted Asarco and completed a reserve study and title evaluation. Application was filed for mining permits and the property was purchased by Nova. In 1985, the property was sold to Inspiration Mines, Inc. (later WestGold), but Nova retained the right of reassignment upon cessation of operations by WestGold. WestGold drilled 2,500 holes, collected 57 bulk samples, and transformed the data from the Shell-Asarco drilling to form one database. WestGold also carried out extensive high-resolution seismic surveys of the lease area. Side scan sonar was used to map sediment types on the seafloor.

4.1.2 Problem of Gold Reserve Estimation

As described above, Shell Mining Company completed approximately 568 drill holes in 1964, and ASARCO drilled an additional 500 holes in 1969. The Shell-ASARCO drilling covered most of the shallower parts of the lease block and formed the basis for preliminary estimates. During the period from the winter of 1987 through the summer of 1989, WestGold and its predecessor, Inspiration Gold, conducted 5 drilling programs and one bulk sampling program, adding an additional 2500 drill holes and 57 bulk samples. The data from 3,500 drill holes and 57 bulk samples were made available by Nova resources (1998) for this analysis.

The lease boundary, excluding the grounds previously dredged, is divided into two categories, "Proven" and "Probable". Proven reserves are limited to areas where drill hole spacing does not exceed 70m x 70m, and probable reserves are generally defined by drilling on 100m x 200m grids. In addition to the proven-probable divisions, the lease area is divided into 9 blocks: COHO, HALIBUT, HERRING, HUMPY, KING, PINK, RED, SILVER, TOMCOD.

These blocks present a significant gold resource in the Nome area that possibly could be mined economically. Initial study was conducted by Ke, 2002 using a neural network (NN) technique for ore reserve estimation. The study did not take into account some of the remedial measures that could be taken to improve the NN model generalization. It is believed that significant risks associated with the ore reserve estimation could be prevented by proper model generalization and validation. However, NN model generalization and validation pose particular difficulties when the drill hole data are sparse and scanty. Researchers (Bowden et al., 2002; Samanta et al., 2004a; Samanta et al., 2004b) have tried various approaches to deal with sparse data by various data division techniques. Random data division is one such method. However, random data division in a sparse data situation might be unreliable. One of the goals of this research is to devise techniques for proper division of the data. In order to improve the generalization of the NN models the entire data set was divided into three subsets: training, calibration and validation, using various techniques. It was assumed that x-coordinate, y-coordinate, water-table depth and gold concentration are input variables. The output of the data division approaches will be three statistically similar subsets which can be used for NN modeling.

4.1.3 Sparse Data Problem

The exploratory boreholes (Table 4.1) provided by Nova Resources are the only available sources of information from which detailed reserve modeling was carried out. Figures 4.3 through 4.11 present the spatial plots of the borehole samples of the nine mining blocks. From the figures, it can be observed that borehole data are sparse for reserve estimation, considering the high spatial variation of ore grade that is commonly associated with placer gold deposit. The boreholes are also not evenly located in the areas. Characterization of the limited borehole data are essential for ore reserve estimation of the subject areas. In order to achieve this goal, descriptive statistical analyses of the data sets were conducted. Table 4.2 shows the mean and the standard deviation values of the Nome gold data sets, which indicate strong variation for all the nine fish blocks. The coefficient of variation is greater than one for all the blocks, which indicates the presence of extreme values in the data sets. Histogram plots of the gold data are also presented in Figure 4.12. The histogram plots illustrate that the gold values are positively skewed. A log-normal distribution may be a suitable fit to the data. Visual portrayal of the histogram plots also reveals that the gold data sets are composed of a large proportion of low values and a small proportion of extremely high values. A closer inspection of the spatial distribution of the high and the low gold grade values also portrays a distinct spatial characteristic of the deposit. For example, the high values do not exhibit any regular trend. Instead, one or two extreme high values occasionally occur in a mix of low values. This may pose a particular difficulty in ore grade modeling since the pattern of occurrence of extreme high values is somewhat unpredictable.

A variography study was also undertaken to analyze the spatial correlation of the gold attributes. Figures 4.13 through 4.21 present the variogram plots of the gold data for all the blocks. The variogram plots exhibit a very small amount of regional component. Large proportions of spatial variability occur from the nugget effect. This indicates the presence of a poor spatial correlation structure in the deposit. Poor spatial correlation, in general, tends to suggest that prediction accuracy for this deposit might not be reliable.

Table 4.1: Number of drill holes in the various mining blocks

Mining Blocks	No of Drill holes
Coho	143
Halibut	323
Herring	415
Humpy	212
King	275
Pink	216
Red	530
Silver	415
Tomcod	450

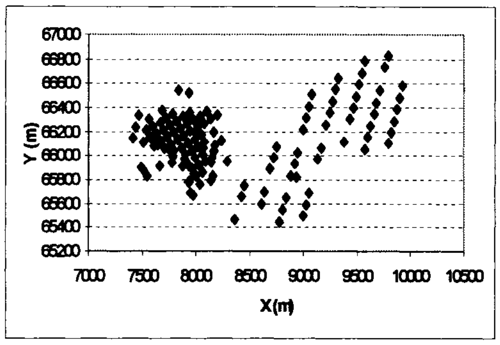


Figure 4.3: Location of drill holes in Coho block

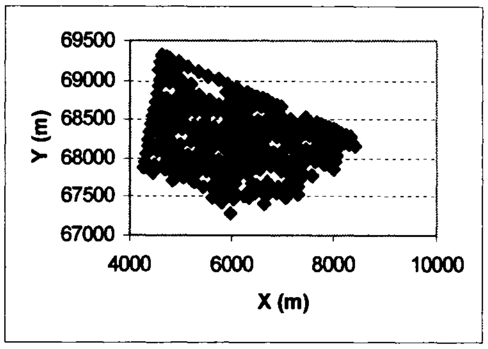


Figure 4.4: Location of drill holes in Halibut block

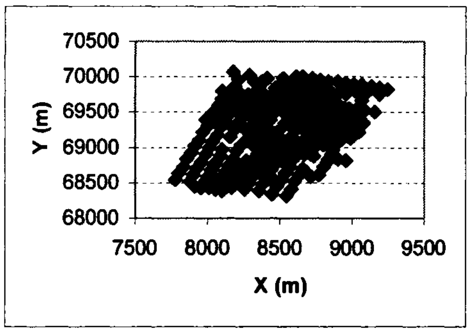


Figure 4.5: Location of drill holes in Herring block

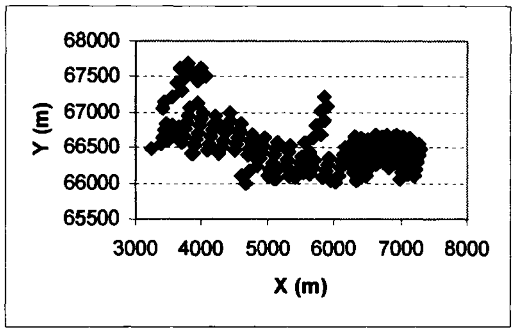


Figure 4.6: Location of drill holes in Humpy block

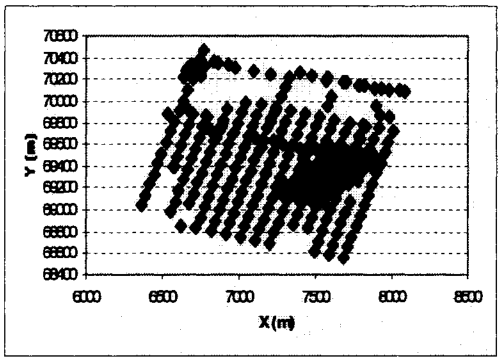


Figure 4.7: Location of drill holes in King block

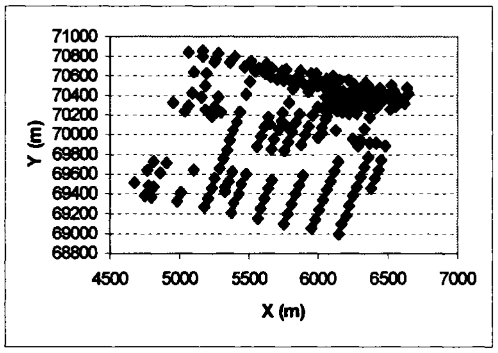


Figure 4.8: Location of drill holes in Pink block

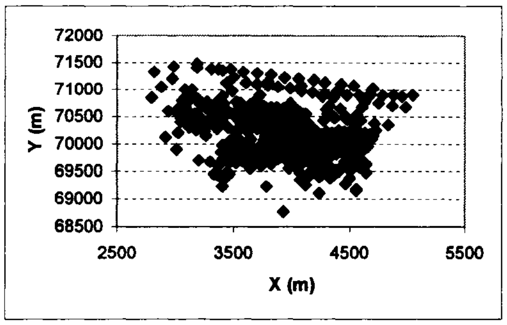


Figure 4.9: Location of drill holes in Red block

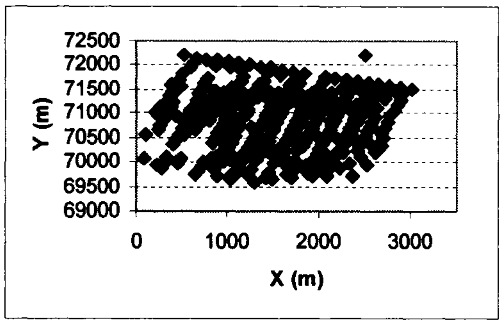


Figure 4.10: Location of drill holes in Silver block

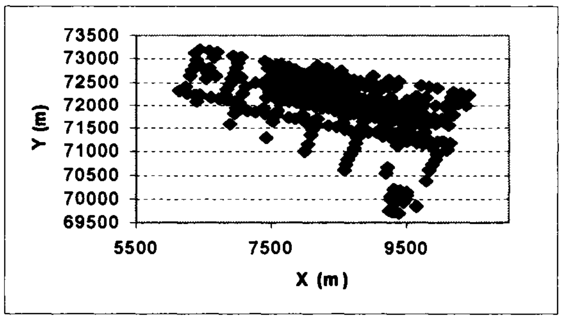


Figure 4.11: Location of drill holes in Tomcod block

Table 4.2: Summary statistics of Nome gold data sets

Data Set	Mean	Standard Deviation	Coefficient of Variation
Coho	620.01	1233.8	1.98
Halibut	241.21	394.52	1.63
Herring	333.35	553.31	1.66
Humpy	449.60	690.64	1.54
King	348.50	665.64	1.91
Pink	125.90	167.74	1.33
Red	440.17	650.58	1.48
Silver	228.36	367.69	1.61
Tomcod	265.31	414.05	1.56

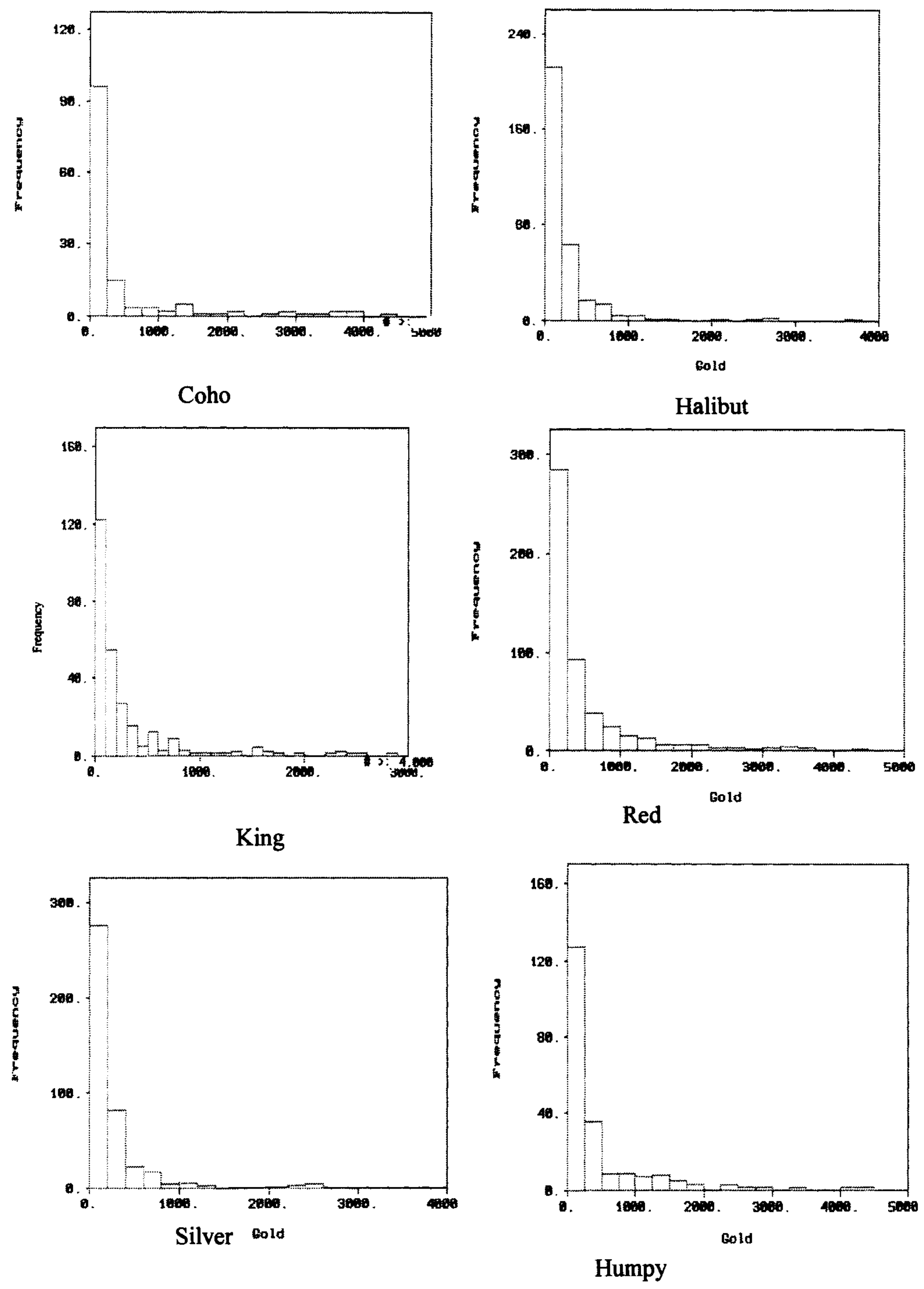


Figure 4.12: Histogram plot of Nome Gold data (various fish blocks)

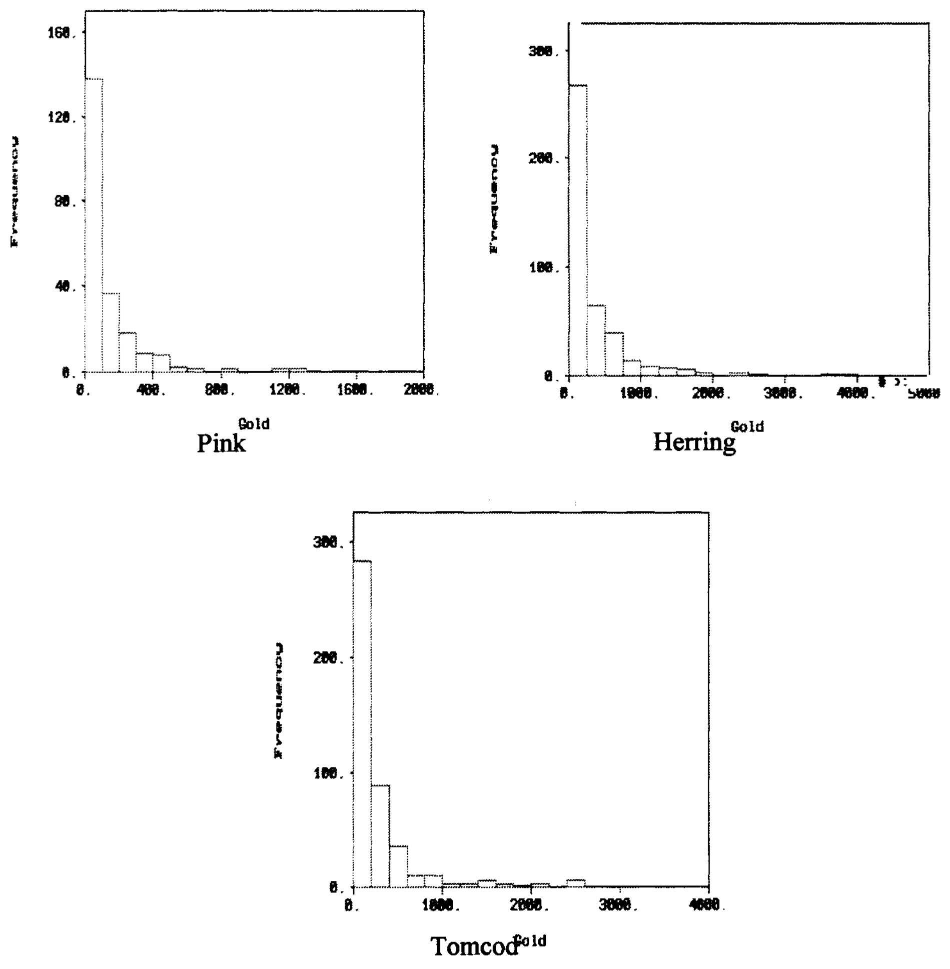


Figure 4.12: Histogram plot of Nome Gold data (various fish blocks)

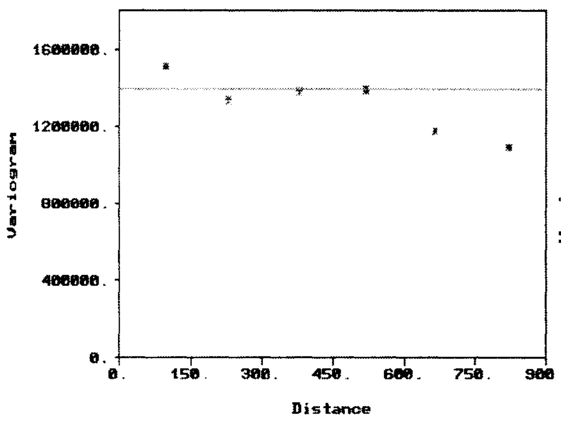


Figure 4.13: Variogram plot of Coho data set

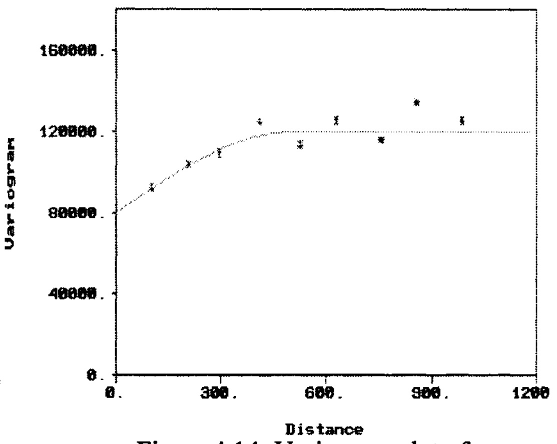


Figure 4.14: Variogram plot of Halibut data set

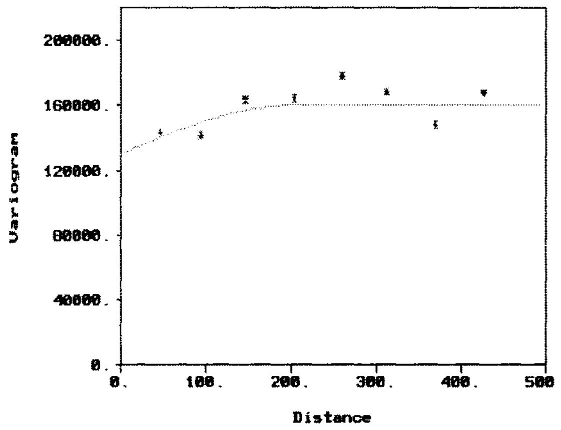


Figure 4.15: Variogram plot of Herring data set

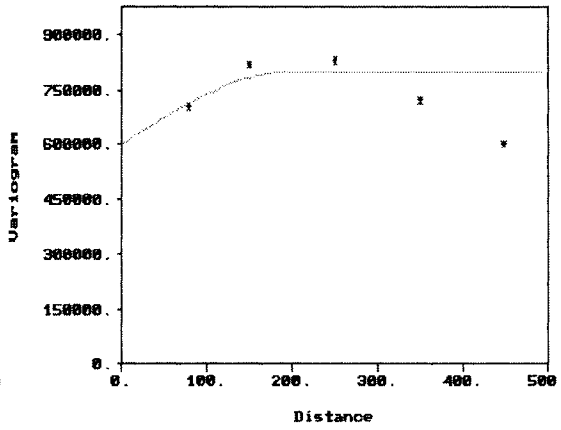


Figure 4.16: Variogram Plot of King data set

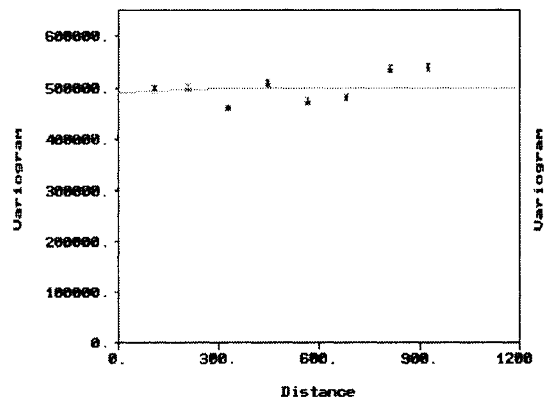


Figure 4.17: Variogram plot of Humpy data set

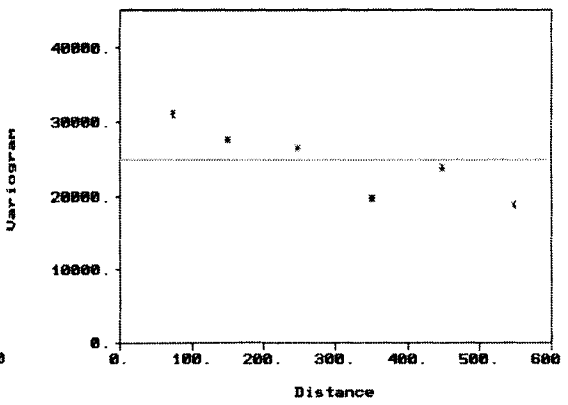


Figure 4.18: Variogram Plot of Pink data Set

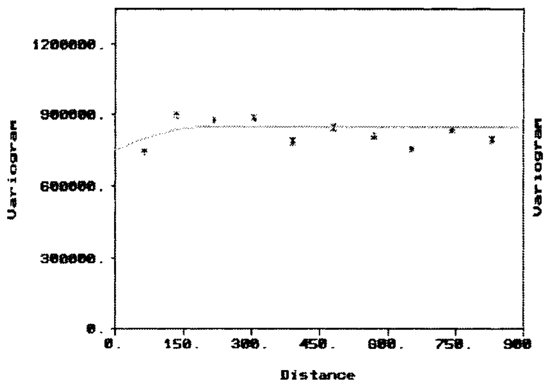


Figure 4.19: Variogram Plot of
Red data set

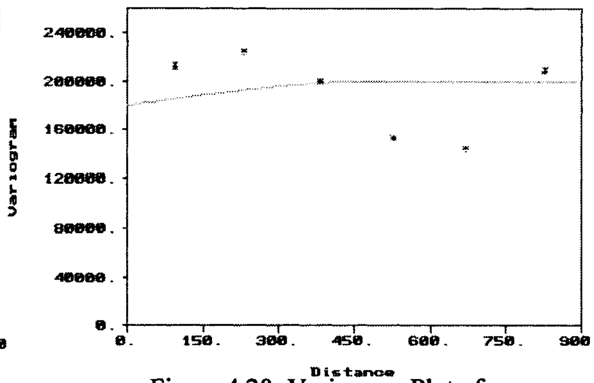


Figure 4.20: Variogram Plot of
Silver data set

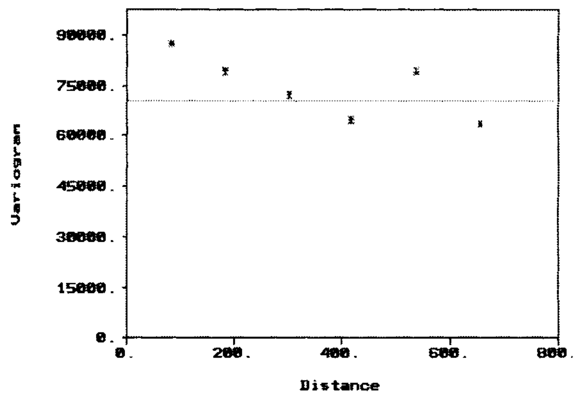


Figure 4.21: Variogram Plot of
Tomcod data set

It is recognized that the available gold data are sparse and exhibit a low level of spatial correlation. Spatial modeling of these data sets is complex. The prediction accuracy may further be reduced if the sparse data problem is not addressed. As described before, random divisions of data may be unrealistic when selecting the members of the sample group into training, calibration and validation subsets. Since the data are characterized by few extreme high values, the training, the calibration and the validation data sets may disproportionately assimilate extreme values if random data division is used.

To demonstrate the severity of data sparseness in random data division, a simulation study was conducted using the Nome data sets. One hundred random data divisions were generated in which sample members for training, calibration and validation subsets were chosen randomly. Statistical similarity tests of the three data subsets, using analysis of variance (ANOVA) and Wald tests, were conducted. All the important attributes (x-coordinate, y-coordinate, water-table depth and gold concentration) were considered in the data division.

Statistical tests such as ANOVA and F tests are most frequently used to compare the means of several population groups. The F test for this study was applied to compare the means of the three data subsets. Therefore, the null hypothesis is: $\mu_T=\mu_C=\mu_v=\mu$, where μ_T, μ_C, μ_v, μ are the means of the training, the calibration, the validation and the entire data sets. The F statistic basically incorporates two factors: (a) within the group variance, and (b) between the group variance. In fact the, F statistic measures the ratio of the variance within the group to the variance between the groups. If the calculated F value exceeds a threshold limit, then the population means are deemed different and the null hypothesis is rejected. The threshold value is usually fixed at a 5% level of significance. This test was used to identify the unacceptable data divisions with respect to each of the variables listed above, namely the x-coordinate, the y-coordinate and the water-table depth, excluding the gold attributes. It is known that the F test often fails to provide a correct interpretation in skewed or log-normal populations. As the gold concentration is better represented by a log-normal distribution, the use of F test may not be advisable. Although log transformation of the data could be used for the F test, the problem, however, was complicated as the gold values were accompanied with some zero observations. Therefore, an F test for log transformed data was also prohibited.

A Wald test was proposed by Tu and Zhou (1999) for log-normal populations with zero values. Therefore, Wald test used for the analysis of the gold data. The Wald test for log-normal distribution with zero observations is based on the idea that the j^{th}

population consists of n_{j0} number of zero observations, and n_{j1} number of nonzero observations, i.e., the total number of observations in the jth population is $n_j=n_{jo}+n_{j1}$. It is assumed that nonzero observations follow lognormal distribution, i.e., log $(Y_{ij})|\ n_{j1} \sim N(\mu_j, \sigma^2_j)$, and zero observations follow a delta distribution, i.e., $Y_{ij}|n_{jo} \sim \text{Bin}\ (n_j, \delta_j)$, where δ_j is the proportion of zero observations in the population. Using a likelihood-based Wald statistic, Tu and Zhou (1999) showed that the Wald statistic, W, has the following form:

$$W=\sum_{j=2}^{K}\frac{n_j\exp(\hat{v}_j)(\hat{v}_1+\hat{\mu}_1+\hat{\sigma}_1^2/2-(\hat{v}_j+\hat{\mu}_j+\sigma_j^2/2))^2}{1-\exp(v_j)+\hat{\sigma}_j^2+\hat{\sigma}_j^4/2}$$

$$-\left(\sum_{j=1}^{K}(n_j\exp(\hat{v}_j))(1-\exp(\hat{v}_j)+\sigma_j^2+\hat{\sigma}_j^4/2)^{-1}\right)^{-1} \tag{4.1}$$

$$\text{x}\left(\sum_{j=2}^{K}\frac{(n_j\exp(\hat{v}_j))(\hat{v}_1+\hat{\mu}_1+\hat{\sigma}_1^2/2-(\hat{v}_j+\hat{\mu}_j+\hat{\sigma}_j^2/2))}{1-\exp(v_j)+\hat{\sigma}_j^2+\hat{\sigma}_j^4/2}\right)^2$$

Under the null hypothesis, the statistic W converges to χ_r^2 with r = K-1 degrees of freedom and where K= number of populations (in this study K=3).

$$\hat{v}_j=\log\frac{n_{j1}}{n_j},\ \mu_j=\frac{1}{n_{j1}}\sum_{i=1}^{n_{j1}}\log y_{ij},\ \text{and}\ \hat{\sigma}_j^2=\frac{1}{n_{j1}}\sum_{i=1}^{n_{j1}}(\log y_{ij}-\hat{\mu}_j)^2$$

Similar to the F test, for the three variables, the calculated value of W was compared for the gold attribute with a threshold limit chosen from the χ^2 table at the 5% level of significance with 2 degrees of freedom. If the calculated W exceeds the threshold limit, then the particular data division was considered bad with respect to the gold attribute.

The results of the simulation study are presented in Table 4.3. Table 4.3 shows the numbers of bad divisions out of 100 data divisions with respect to each of the variables, both separately and jointly. The numbers presented in Table 4.3 are the average values of the one hundred simulation runs, where each run consisted of 100 random data divisions. It may be observed that a large number of unacceptable data divisions could result if random data division is employed. For example, the Coho data set's numbers of bad data divisions (out of one hundred) are 6, 6, 6 and 13 for the variables x-coordinate, y-coordinate, water-table depth and gold concentration respectively. Overall, a total of 27 data divisions out of 100 are bad when all the variables are considered together. Furthermore, a particular data division might be bad with respect to one or more variables. Therefore, the numbers of bad divisions with respect to the individual variables do not sum up to the total number of bad divisions when all the variables are considered together.

It may be obvious that almost ¼ of the data divisions are bad in random data divisions due to the existing sparseness. This can be regarded as quite significant. Unreliability of the random data division is further explored through the inspection of bad data divisions. Statistical summaries for one of the arbitrarily selected random data divisions for all the nine data sets are presented in Tables 4.4 through 4.12. From the tables, it is seen that both the mean and the standard deviation values are significantly different among the data subsets for the blocks considered.

Table 4.3: Number of bad divisions out of 100 random divisions

Data set	Xcordinate	Ycordinate	Water Table Depth	Gold	Total
Coho	6	6	6	13	27
King	6	6	6	17	28
Halibut	6	7	6	15	29
Humpy	6	6	6	9	24
Red	6	6	6	9	22
Tomcod	6	5	5	22	32
Pink	6	6	6	13	24
Herring	6	6	6	11	27
Silver	6	6	6	7	19

Table 4.4: Statistical summary of one of the random divisions for the Coho data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	8365.5	8661.5	8045.1	8100.5	724.16	847.2	362.71	451.47
Y	66119	66281	66040	65883	271.8	208.39	243.78	189.83
Gold	620.01	868.16	274.07	471.08	1233.8	1608.6	315.43	786.54
WTD	19.184	19.137	18.566	19.859	1.0061	0.8050	1.2693	0.6048

Table 4.5: Statistical summary of one of the random divisions for the Halibut data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	6193.5	6153.9	6312.6	6151.2	1046.4	1034.9	1154.7	952.62
Y	68258	68243	68370	68172	412.7	453.9	386.0	321.35
Gold	241.21	189.64	131.78	457.23	394.52	212.0	112.07	682.24
WTD	15.603	15.653	15.301	15.81	1.3069	1.3337	1.2925	1.2262

Table 4.6: Statistical summary of one of the random divisions for the Herring data set

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	8507	8503.5	8436.3	8586.2	307.92	290.52	319.28	314.54
Y	9257.6	9228.7	9173.8	9401	440.03	474.5	447.17	312.25
Gold	333.35	228.47	246.02	633.1	553.31	269.91	285.99	944.00
WTD	8.1429	8.324	9.1762	6.72	4.408	4.61	4.4039	3.603

Table 4.7: Statistical summary of one of the random divisions for the Humpy data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	5624.7	5301.6	5788.9	6106.7	1225.4	1293	1148	960.2
Y	66527	66626	66513	66343	325.8	363.86	286	163.47
Gold	449.6	243.18	264.28	1047.8	690.64	345.15	0.653	1054.3
WTD	18.392	18.331	18.192	18.710	0.930	0.94	311.25	1.0760

Table 4.8: Statistical summary of one of the random divisions for the King data set

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	177260	177150	177190	177530	423.73	373.53	496.23	307.5
Y	1169500	1169500	1169600	1169400	429.21	501.21	391.41	270.05
Gold	348.55	124.05	224.00	908.93	665.64	129.49	266.14	1106.2
WTD	8.293	8.78	7.28	8.31	3.94	4.48	3.60	2.84

Table 4.9: Statistical summary of one of the random divisions for the Pink data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	5839.1	5925.2	6039	5467	479.38	399.68	396.28	505.73
Y	10092	10033	10388	9914.1	467.16	469.57	195.25	523.12
Gold	125.98	180.06	71.589	72.20	167.74	215.32	71.840	58.840
WTD	6.022	6.242	3.264	8.340	3.9780	3.9290	1.2280	4.2600

Table 4.10: Statistical summary of one of the random divisions for the Red data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	3941.8	3947.7	3838.2	4032.6	456.54	436.89	505.50	425.5
Y	10198	10174	10350	10097	469.75	483.19	487.22	384.06
Gold	440.17	297.75	781.77	385.00	650.58	353.31	10340	475.12
WTD	8.4845	8.94	6.6242	9.414	5.2063	5.1679	5.3559	4.69

Table 4.11: Statistical summary of one of the random divisions for the Silver data set

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	11511	11501	11387	11655	682.3	600.32	697.23	791.16
Y	10766	10721	10488	11127	612.96	571.75	491.16	632.22
Gold	228.36	318.56	168.11	109.62	367.69	483.44	169.90	113.53
WTD	10.305	10.716	12.646	7.200	4.7248	4.4091	3.2167	4.9710

Table 4.12: Statistical summary of one of the random divisions for the Tomcod data set

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	8508.2	8378.2	9002.9	8276.7	917.61	837.73	727.59	1055.7
Y	12010	12116	11501	12304	630.27	419.93	832.78	428.33
Gold	265.31	171.02	534.66	186.07	414.09	177.61	698.24	210.66
WTD	7.2951	6.990	9.7268	5.4912	3.9186	3.321	4.5255	3.133

4.1.4 Data Segmentation for Data Division

From the preceding discussion related to the sparse data, it is obvious that random data divisions are not satisfactory, particularly for gold attributes. Statistical indices, as well as the histogram plots presented earlier, indicated that the gold data are highly skewed and follow a log-normal distribution. There is considerable evidence that the gold distribution is represented by a few extreme high values, and that the presence or the absence of these extreme values in a data subset significantly influences the statistical characteristics of that subset. Since the spatial pattern of the few extreme high values is quite different from the low values, special attention should be paid so that extreme values are selected equally into the three data subsets. Furthermore, low and medium range values must also be present in the right proportion in these three data sets.

It is likely that the use of a genetic algorithm or kohonen network may not result in equal representation of the low, the medium and the high value patterns in the three data sets. Hence, prior to the application of a GA or Kohonen network for data division, data segmentation for low, medium and high range values should be used to improve the results. This assertion is supported through a study of data division using a GA or kohonen network with data segmentation and without data segmentation. A statistical summary of the data division without data segmentation using a genetic algorithm or kohonen network is presented in Table 4.13 for the King data set. From the table, it is seen that although the GA and kohonen networks appreciably improve the consistency of the data subsets, this could further be improved using data segmentation. Therefore, data sets were partitioned into three segments before applying the GA or kohonen network in the individual segments.

Table 4.13 (a): Genetic algorithm for data division in the King data set (without data segmentation)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	177260	177264	177274	177224	423.7	425.8	421.9	425.6
Y	1169500	1169497	1169471	1169510	429.2	430.9	433.6	426.6
Gold	348.55	365.0	333.6	330.8	665.6	691.3	649.1	636.6
WTD	8.293	8.24	8.53	8.51	3.94	3.95	3.97	3.96

Table 4.13 (b): Kohonen network for data division in the King data set (without data segmentation)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	177260	177270	177240	177260	423.7	442.7	418.2	410
Y	1169500	1169500	1169500	1169500	429.2	453.8	420.8	411
Gold	348.55	425.3	318.6	289.40	665.6	831.1	586.3	503
WTD	8.293	8.25	8.52	8.10	3.94	4.05	3.94	3.86

4.1.4.1 Data Segmentation and GA for Data Division

Prior to applying GA, the data were divided into three prime segments. The segmentation was done based on a visual inspection of a histogram plot. Figure 4.22 presents a schematic diagram of the data segmentation and the genetic algorithm approach. After the data segmentation, GA was applied in each of the segments: segment 1, segment 2 and segment 3. A MATLAB program was developed for the GA application in data division. The members of the training, calibration and the validation data sets were selected using GA from each of the segments. Once the members for the training, calibration and validation data were chosen, the selected members from the segments were appended together to form the respective subsets. Tables 4.14 through 4.22 present the summary statistics of the mean and the standard deviation values for all the variables for the three data subsets and entire data set. It is observed that the mean and standard deviation values are similar for all the data subsets. The histogram plots of the three subsets and the entirety of the nine data sets are presented in Figures 4.23 through 4.31. From the figures, it is seen that all the data subsets assume an almost identical shape to that of the overall data set, and that the skewness of the data in the three subsets is also preserved.

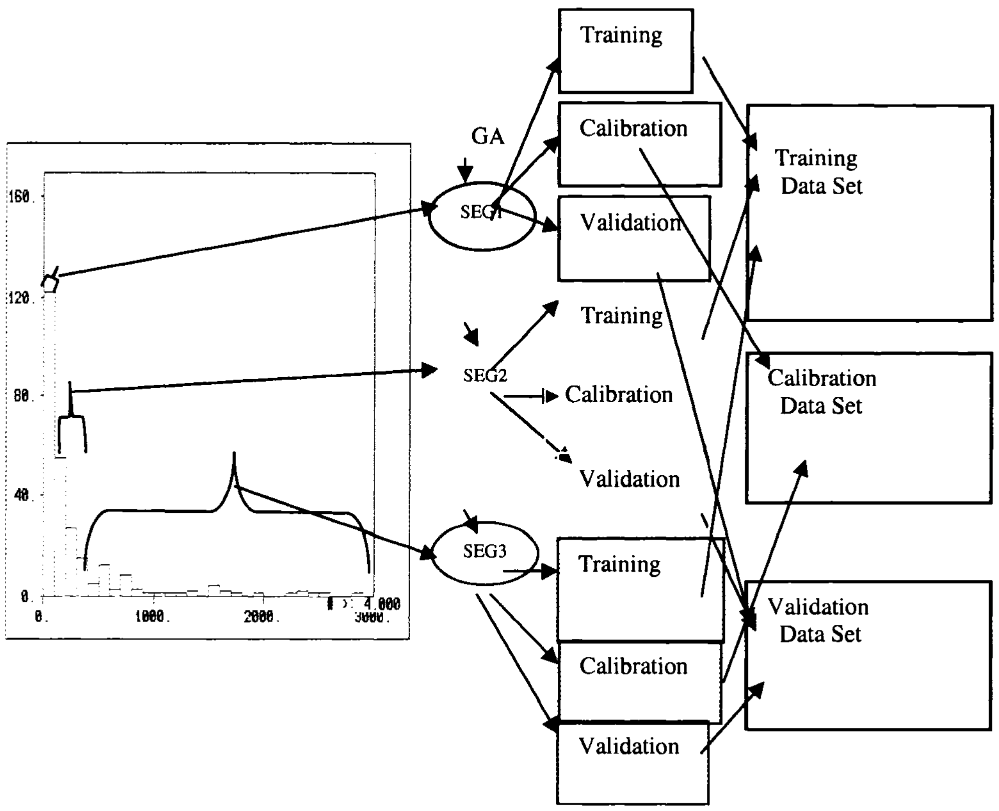


Figure 4.22: Data segmentation and genetic algorithms for data divisions

Table 4.14: Statistical summary of data division using GA (Coho)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	8365.5	8396.7	8334.2	8335.2	724.16	727.74	732.61	726.45
Y	66119	66125	66113	66112	271.8	274.06	272.64	273.83
Gold	620.01	626.63	611.65	615.30	1233.8	1265	1214.1	1225.3
WTD	19.184	19.03	19.32	19.33	1.0061	1.07	0.90	0.94

Table 4.15: Statistical summary of data division using GA (Halibut)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	6193.5	6193.0	6226.5	6162.4	1046.4	1049.4	1047.9	1051.1
Y	68258	6825.4	6827.5	6824.7	412.7	410.7	418.40	415.49
Gold	241.21	239.18	233.73	252.48	394.52	382.74	354.04	454.93
WTD	15.603	15.62	15.59	15.57	1.3069	1.29	1.32	1.32

Table 4.16: Statistical summary of data division using GA (Herring)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida-tion
X	8507	8505.4	8509.6	8507.5	307.92	308.4	309.3	308.4
Y	9257.6	9256.9	9256.4	9260.3	440.03	443.15	441.49	436.46
Gold	333.35	324.04	350.24	334.43	553.31	561.09	558.38	536.83
WTD	8.1429	8.07	8.21	8.20	4.408	4.44	4.40	4.37

Table 4.17: Statistical summary of data division using GA (Humpy)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Validation
X	5624.7	5647.8	5611.9	5592.7	1225.4	1232	1208.7	1249.7
Y	66527	66529	66519	66530	325.8	331.37	319.12	327.16
Gold	449.6	455.90	460.21	427.53	690.64	713.70	680.48	666.90
WTD	18.392	18.38	18.38	18.41	0.93	0.94	0.93	0.94

Table 4.18: Statistical summary of data division using GA (King)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida-tion
X	177260	177240	177270	177270	423.73	431.29	416.07	421.23
Y	1169500	1169500	1169500	1169500	429.21	430.23	432.20	429.51
Gold	348.55	337.44	366.27	353.25	665.64	689.16	639.66	652.07
WTD	8.293	8.436	8.170	8.132	3.94	3.951	4.024	3.915

Table 4.19: Statistical summary of data division using GA (Pink)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Validation
X	5839.1	5829.6	5844.5	5852.2	479.38	481.9	485.46	477.01
Y	10092	10100	10092	10077	467.16	466.16	470.90	473.59
Gold	125.98	127.21	127.10	122.55	167.74	165.28	181.54	161.74
WTD	6.022	6.033	6.030	5.994	3.978	4.01	3.98	3.96

Table 4.20: Statistical summary of data division using GA (Red)

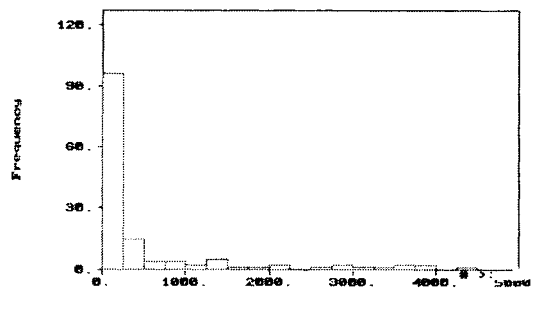
Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Validation
X	3941.8	3950	3935.6	3931.6	456.54	456.6	457.6	458.6
Y	10198	10194	10218	10186	469.75	471.2	467.2	472.4
Gold	440.17	461.99	418.46	418.59	650.58	673.97	627.89	628.8
WTD	8.4845	8.38	8.63	8.54	5.2063	5.23	5.19	5.19

Table 4.21: Statistical summary of data division using GA (Silver)

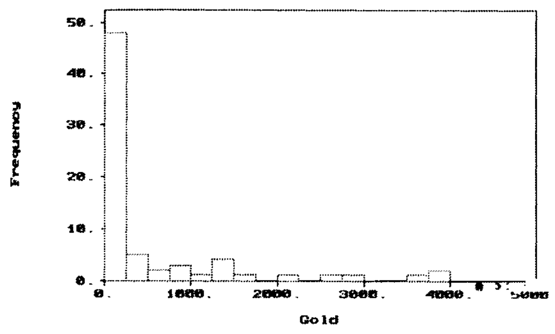
Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	11511	11531	11507	11478	682.3	686.05	683.60	678.78
Y	10766	10766	10769	10763	612.96	616.36	614.43	610.70
Gold	228.36	229.12	219.27	235.53	367.69	375.16	335.18	385.56
WTD	10.305	10.31	10.30	10.28	4.7248	4.73	4.72	4.74

Table 4.22: Statistical summary of data division using GA (Tomcod)

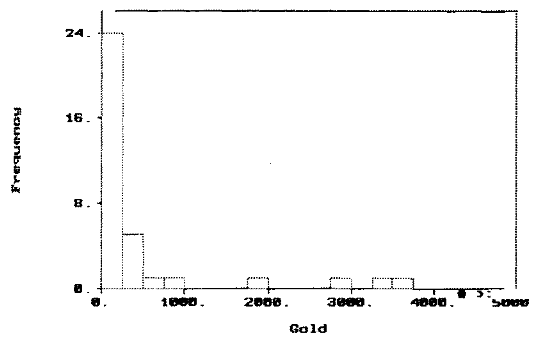
Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	8508.2	8518.8	8488.1	8506.9	917.61	920.86	920.84	915.70
Y	12010	12024	11983	12010	630.27	631.83	634.44	627.74
Gold	265.31	271.22	256.08	262.66	414.09	417.61	410.65	413.8
WTD	7.2951	7.17	7.47	7.35	3.9186	3.93	3.92	3.89



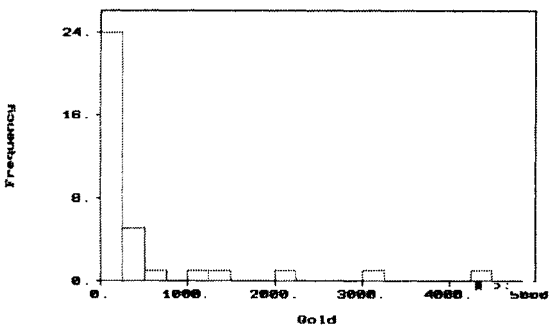
Gold
Entire data set



Training data set

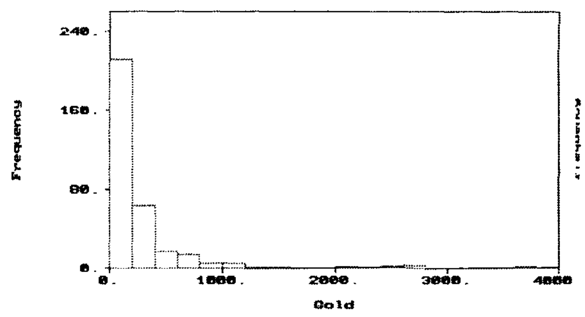


Calibration data set

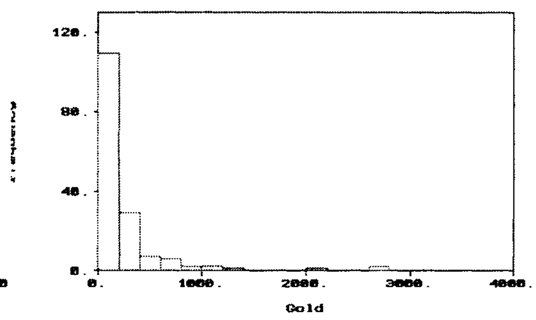


Validation data set

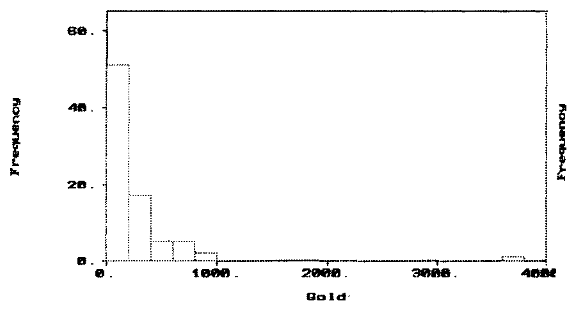
Fig 4.23: Histogram plot of Coho dataset



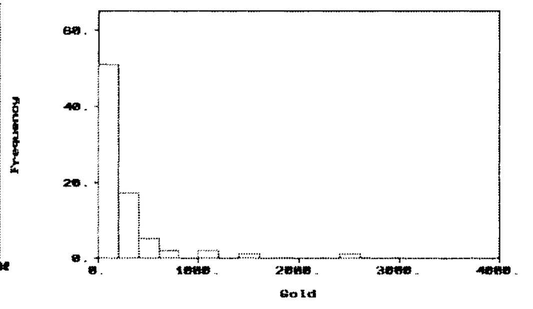
Entire data set



Training data set



Calibration data set



Validation data set

Fig 4.24: Histogram plot of Halibut data set

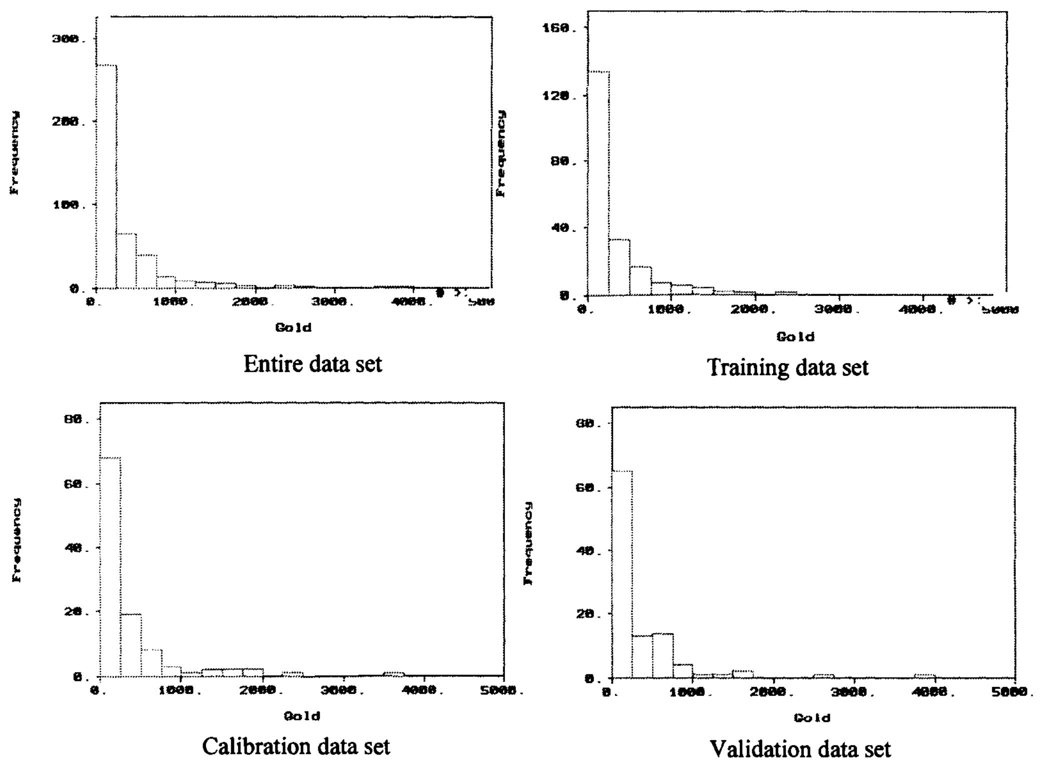


Fig 4.25: Histogram plot of Herring data set

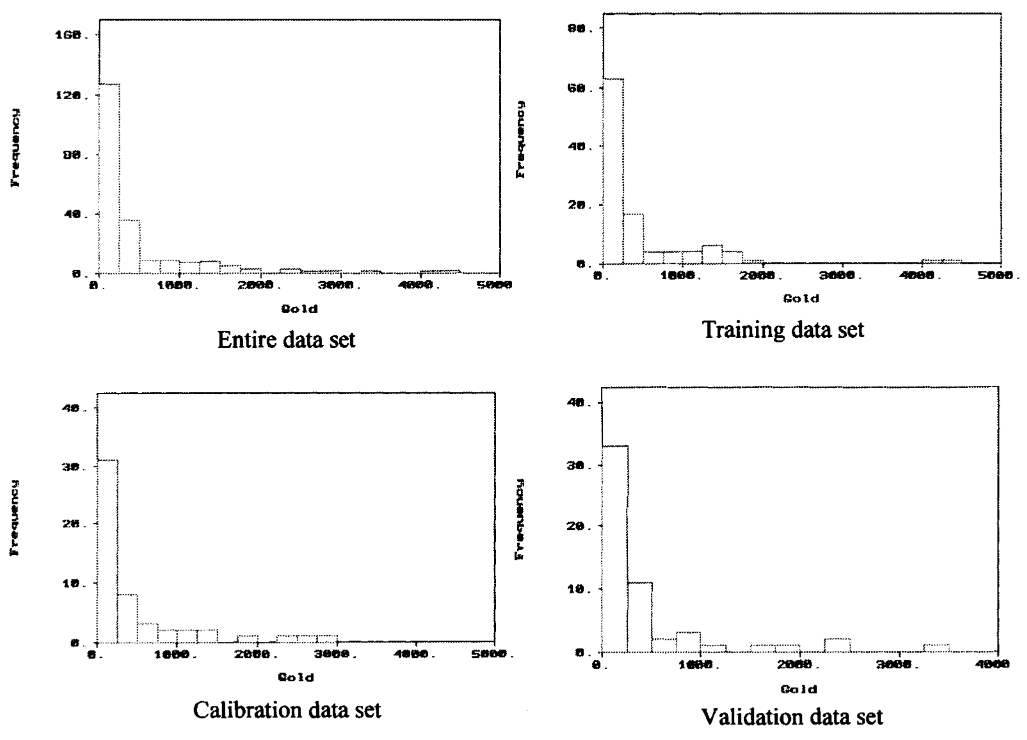


Fig 4.26: Histogram plot of Humpy data set

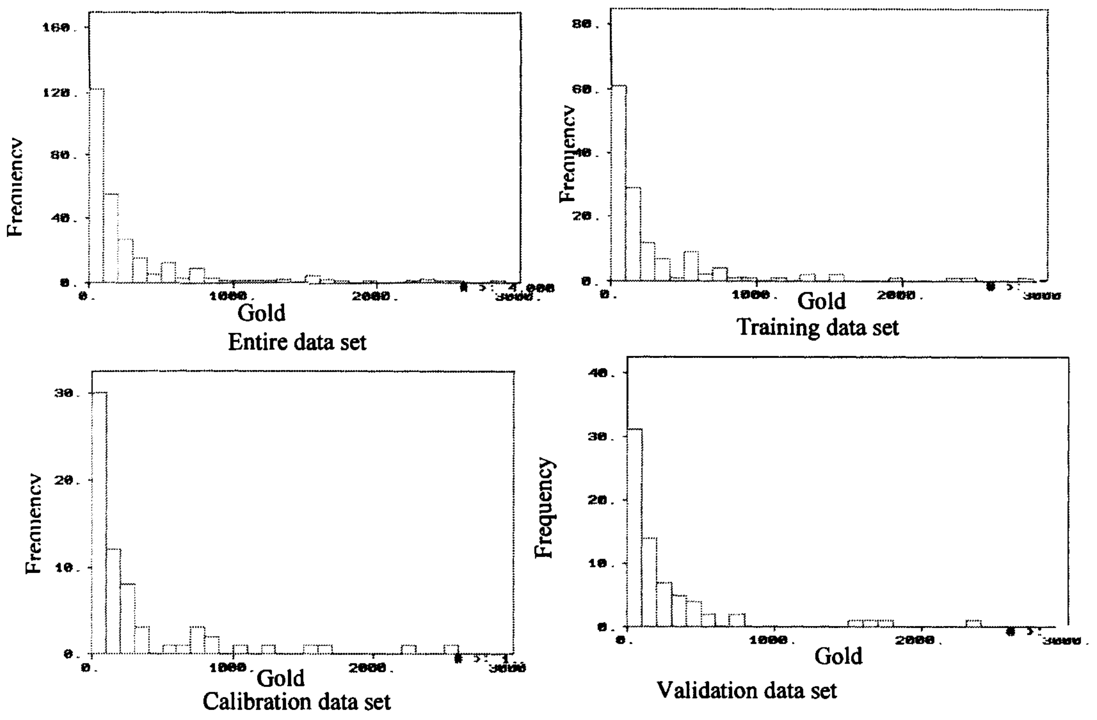


Fig 4.27: Histogram plot of King data set

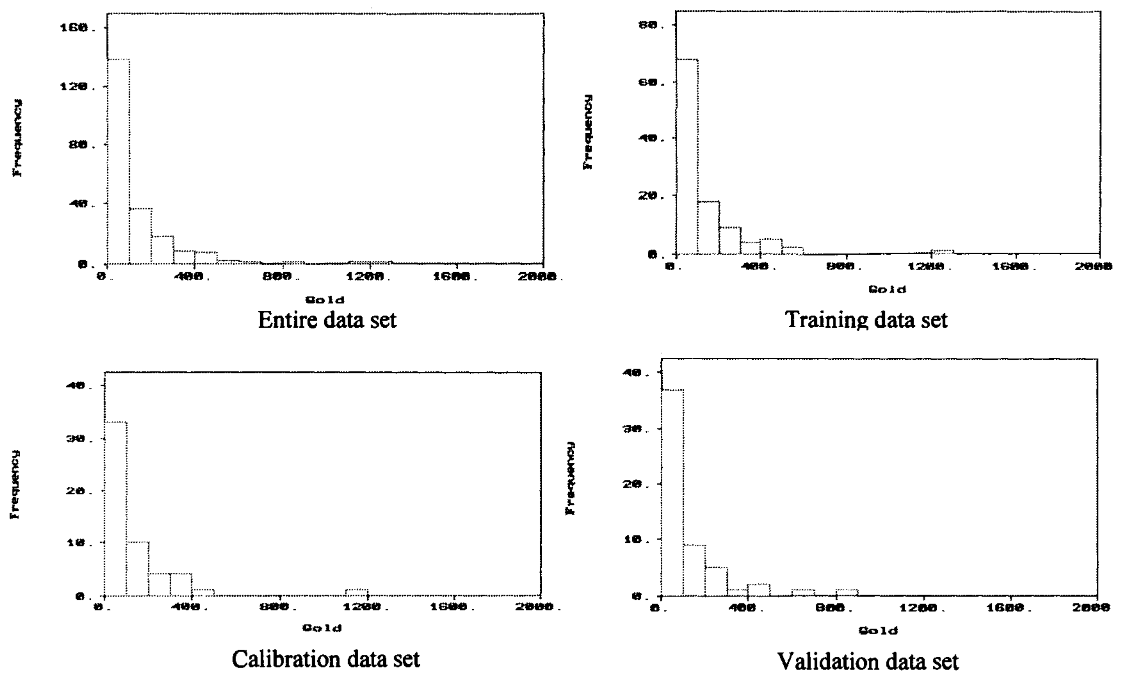
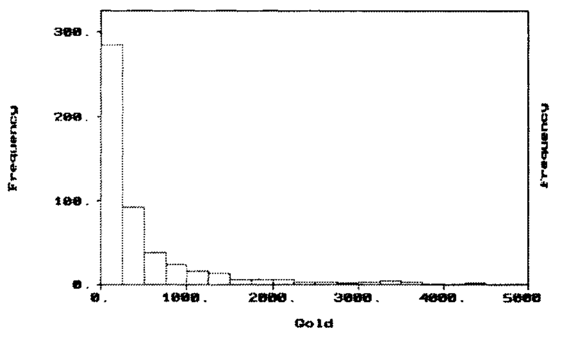
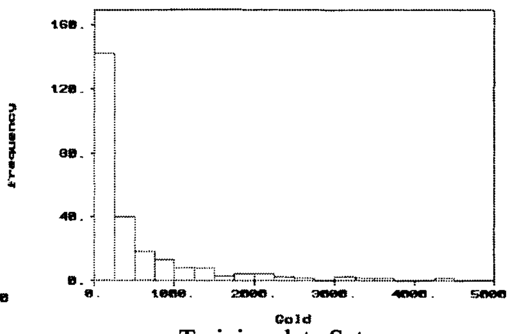


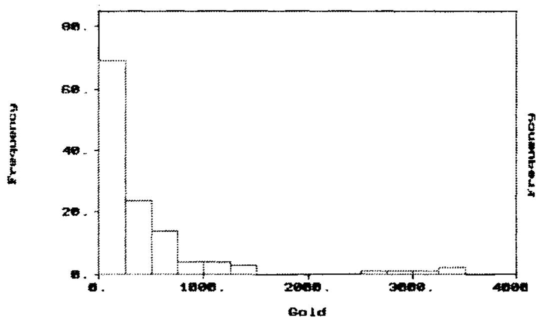
Fig 4.28: Histogram plot of Pink data set



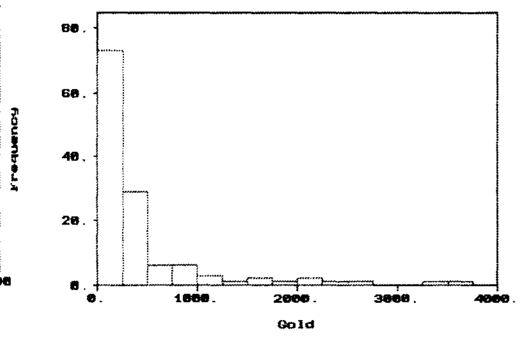
Entire data set



Training data Set

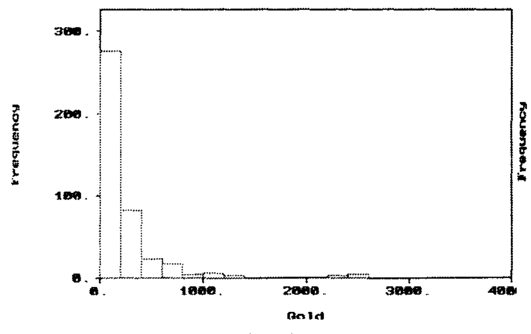


Calibration data set

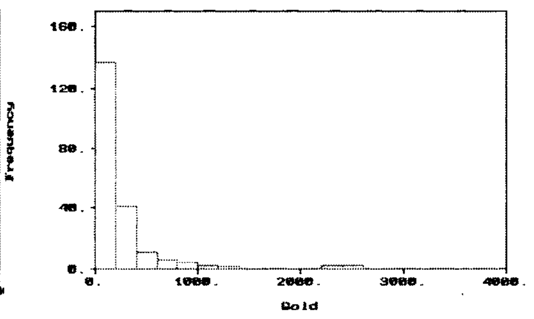


Validation data set

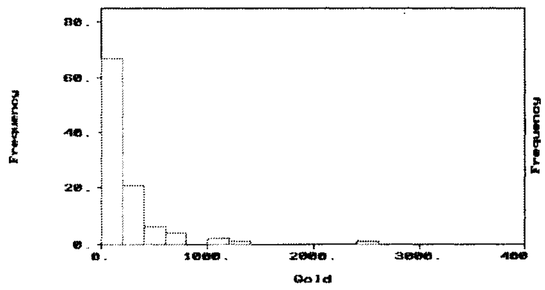
Fig 4.29: Histogram plot of Red data set



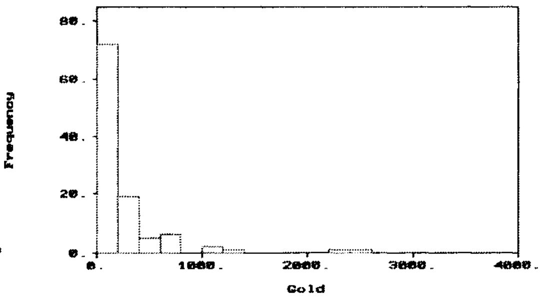
Entire data set



Training data set



Calibration data set



Validation data set

Fig 4.30: Histogram plot of Silver data set

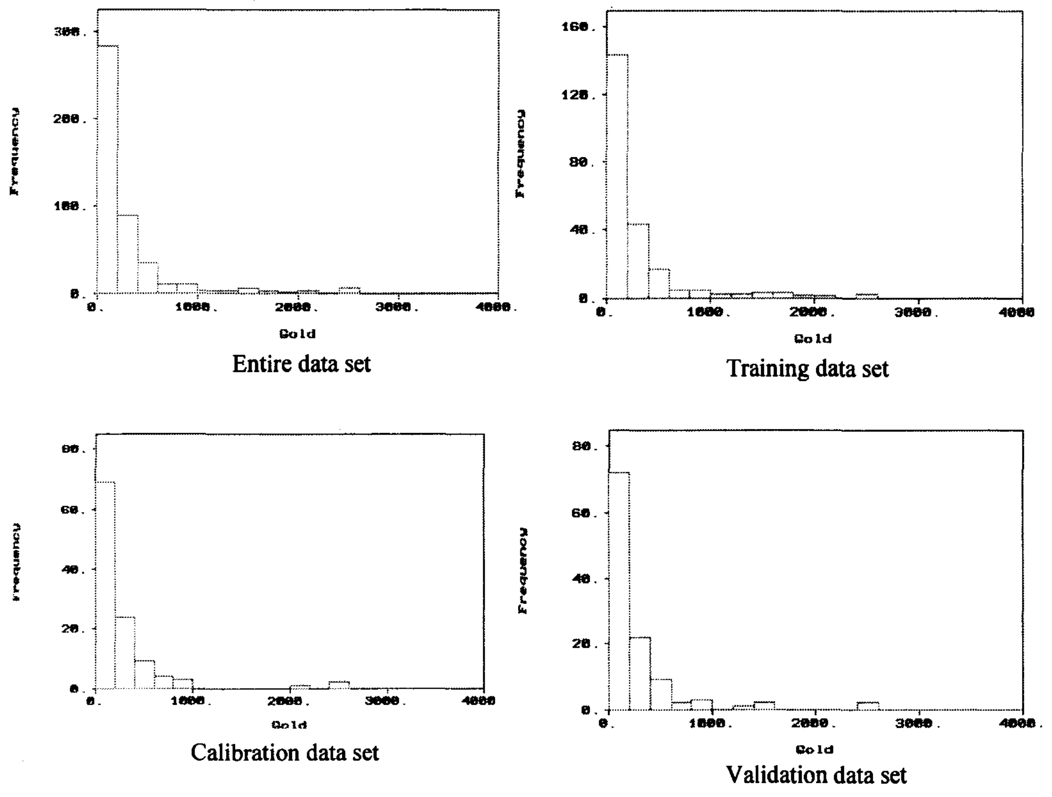


Fig 4.31: Histogram plot of Tomcod data set

4.1.4.2 Data Segmentation and Kohonen Network for Data Division

A kohonen network was also applied in each of the three segments to sample for the training, the calibration and the validation data sets. It was done to verify if improved results can be accrued using this approach. Figure 4.32 presents a schematic diagram of data segmentation and the kohonen network approach. The intent of the kohonen network was to apply an algorithm in each of the segments separately for grouping of the similar patterns in the segment. For this purpose, Neuroshell software (1993) was used. Since the number of clusters needed is not previously known, a large number of clusters, e.g 15, were chosen. Once the grouping were done, each group contained similar patterns. From each group, samples were selected randomly for training, calibration and validation. If there was only one sample in a group, that sample was put into the training set. If there were only two samples, one sample was put into the training, and the other in the validation set. Continuing in this way, the samples were selected from all three segments and the training, the calibration and the validation data sets were prepared. Tables 4.23-4.31 present the summary statistics of the mean and standard deviation values for all the variables of the three data subsets and the entire data set. It may be observed that the mean and the standard deviation values are similar for each of the datasets. In comparison with the genetic algorithm, however, this approach provided slightly inferior results. Despite this outcome, the experimentation showed that data segmentation using the genetic algorithm and kohonen networks is appropriate for division of the data.

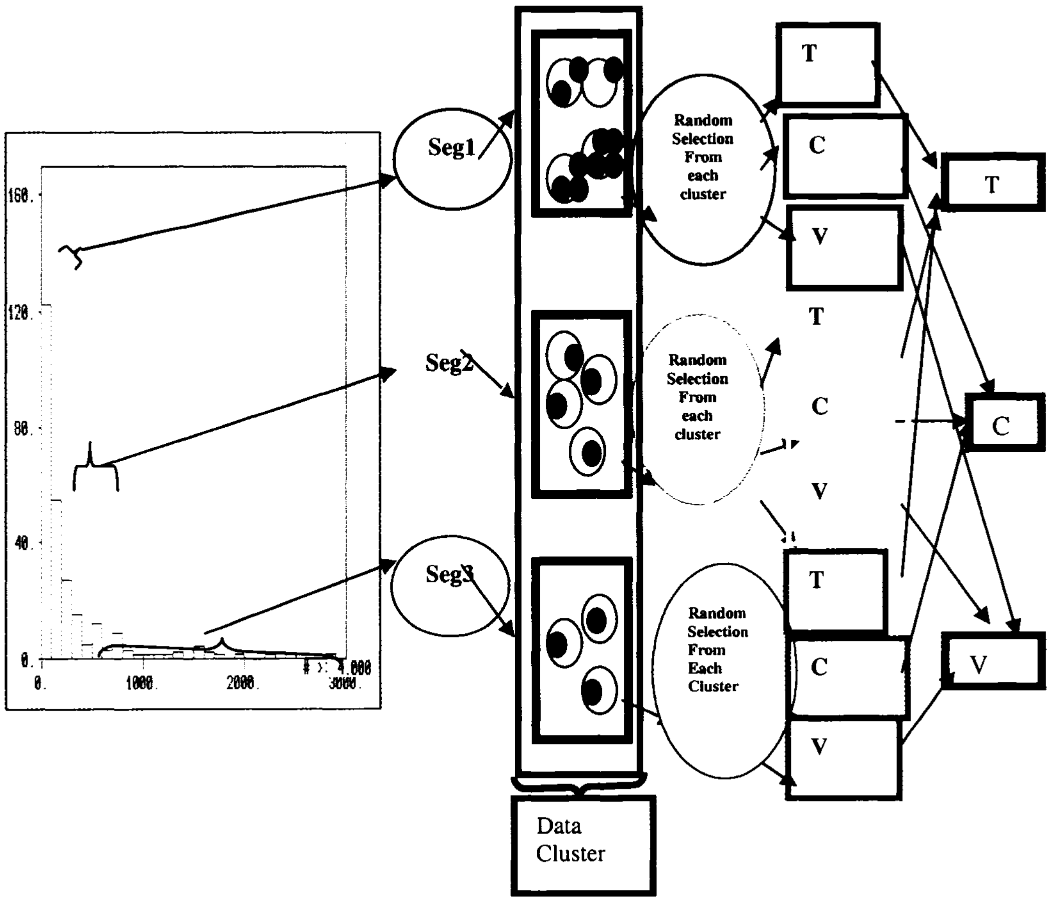

160.
120.
80.
40.
0.
0.
1000.
2000.
3000.
Seg1
Seg2
Seg3
Random Selection From each cluster
Random Selection From each cluster
Random Selection From Each Cluster
T
C
V
T
C
V
T
C
V
T
C
V
Data Cluster

Figure 4.32: Data segmentation and kohonen network for data division

Table 4.23: Summary of the data division using Kohonen network (Coho)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	8365.5	8317.40	8479.20	8377.30	24.16	675.45	788.04	790.47
Y	66119	66083	66202	66129	271.8	265.82	260.52	287.74
Gold	620.01	681.32	597.43	482.42	1233.8	1198.6	1498.50	1048.1
WTD	19.184	19.219	18.929	19.348	1.0061	1.0988	0.80216	0.911

Table 4.24: Summary of the data division using Kohonen network (Halibut)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	6193.5	6121.7	6285.50	6209.60	1046.4	1011.3	1060.90	1086.8
Y	68258	68232	68264	68290	412.7	422.34	400.82	411.73
Gold	241.21	268.58	220.27	220.96	394.52	470.40	324.70	328.86
WTD	15.603	15.701	15.612	15.444	1.3069	1.3322	1.2968	1.2771

Table 4.25: Summary of the data division using Kohonen network (Herring)

Attri- bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	8507	8509.50	8500.50	8510	307.92	314.17	300.64	308.54
Y	9257.6	9269.70	9269.70	9228.20	440.03	434.65	435.61	454.15
Gold	333.35	355.78	328.29	306.35	553.31	629.10	565.12	409.99
WTD	8.1429	8.0231	8.2033	8.2537	4.408	4.3273	4.4979	4.4664

Table 4.26: Summary of the data division using Kohonen network (Humpy)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	5624.7	5531	5739.80	5683.30	1225.4	1224.6	1192.40	1267.7
Y	66527	66533	66517	66526	325.8	340.14	327.57	301.55
Gold	449.6	371.22	540.45	504.11	690.64	586.24	849.30	690.38
WTD	18.392	18.415	18.353	18.391	0.93	0.8519	0.97084	1.0561

Table 4.27: Summary of the data division using Kohonen network (King)

Attri- bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	177260	177250	177260	177270	423.7	414.1	436.6	428.9
Y	169500	1169500	1169500	1169500	429.2	417.2	424.7	454.5
Gold	348.55	365.9	328.3	344.6	665.6	771.2	590.4	579.1
WTD	8.293	8.04	8.56	8.36	3.94	3.74	4.18	4.01

Table 4.28: Summary of the data division using Kohonen network (Pink)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	5839.1	5844.10	5835.90	5834.90	479.38	486.39	480.56	475.44
Y	10092	10097	10076	10102	467.16	478.47	443.91	479.82
Gold	125.98	139.08	125.96	106.56	167.74	189.65	179.85	111.92
WTD	6.022	5.9239	6.1258	6.0661	3.978	3.9213	3.9322	4.1675

Table 4.29: Summary of the data division using Kohonen network (Red)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	3941.8	3937	3928.10	3961.80	456.54	459.96	450.13	460.72
Y	10198	10180	10222	10198	469.75	458.43	451.35	503.46
Gold	440.17	478.90	400.76	428.46	650.58	708.20	5.1278	630.42
WTD	8.4845	8.6561	8.2953	8.4473	5.2063	5.2216	589.52	5.291

Table 4.30: Summary of the data division using Kohonen network (Silver)

Attri -bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida tion
X	11511	11516	11492	11523	686.05	647.86	709.37	708.59
Y	10766	10754	10767	10783	616.36	633.36	604.14	595.88
Gold	228.36	252.49	212.24	209.28	375.16	387.73	356.55	349.18
WTD	10.305	10.394	10.283	10.197	4.73	4.8336	4.6563	4.6688

Table 4.31: Summary of the data division using Kohonen network (Tomcod)

Attri-bute	Mean				Standard Deviation			
	Overall	Training	Calibration	Validation	Overall	Training	Calibration	Valida -tion
X	8508.2	8499.7	8518.8	8508.4	917.61	912.19	947.17	900.99
Y	12010	12025	11987	12014	630.27	675.65	629.73	571.58
Gold	265.31	268.56	264.0	262.43	414.09	422.28	424.97	394.98
WTD	7.2951	7.1114	7.481	7.3453	3.9186	4.0781	3.8627	3.7807

CHAPTER V

NOME GOLD RESERVE ESTIMATION USING ORIGINAL FISH BLOCKS

The Nome gold reserve estimation is challenging owing to the complexity associated with the placer gold deposits, in addition to the scarcity and sparseness of the drill hole data. The drill hole samples are located on an irregular grid and sparsely scattered throughout the region. As described earlier, the lease boundary is divided into 9 blocks viz., Coho, Halibut, Herring, Humpy, King, Pink, Red, Silver and Tomcod for the purpose of grade and reserve estimation.

Each drill hole information contains Northing (Y-coordinate), Easting (X-coordinate), water-table depth, gold grade in mg/m^3, along with other relevant information. An example of drill hole data for the King mining block is presented in Table 5.1. For grade estimation, Northing, Easting and water-table depth were considered as input variables, and the gold grade was considered as an output variable. The composite values of the gold grade, up to a 5m sea-floor depth, were considered in this study. The total number of holes drilled in each fish block is presented in Table 4.1 and the total number of patterns in the validation dataset for all the blocks is shown in Table 5.2.

In this chapter, the gold grade associated in the nine blocks of the Nome deposit has been estimated using the traditional geostatistical ordinary kriging (OK) technique and two machine learning algorithms namely, the neural network (NN) method and the support vector machine (SVM) method. The various issues involved in the use of these techniques for grade estimation are discussed in the following sections.

Table 5.1: Example of drill hole data for King block

DHID	XSPM	YSPM	W_D	IBEGIN	IEND	METER	DRIVE	MGCM	G_T	CUMMG	CUMMG/CM
343	178042.4	1169828.5	2.7	-2.74	-3.87	1.13	1.13	216	244.08	244.08	216.00
343	178042.4	1169828.5	2.7	-3.87	-5.09	2.35	1.22	105	128.1	372.18	158.37
343	178042.4	1169828.5	2.7	-5.09	-6.31	3.57	1.22	105	128.1	500.28	140.13
343	178042.4	1169828.5	2.7	-6.31	-7.53	4.79	1.22	53	64.66	564.94	117.94
343	178042.4	1169828.5	2.7	-7.53	-9.97	7.22	2.44	0	0	564.94	78.25
343	178042.4	1169828.5	2.7	-9.97	-11.19	8.44	1.22	231	281.82	846.76	100.33
343	178042.4	1169828.5	2.7	-11.19	-12.41	9.66	1.22	105	128.1	974.86	100.92
343	178042.4	1169828.5	2.7	-12.41	-14.84	12.1	2.44	0	0	974.86	80.57
343	178042.4	1169828.5	2.7	-14.84	-16.06	13.32	1.22	32	39.04	1013.90	76.12
343	178042.4	1169828.5	2.7	-16.06	-17.28	14.54	1.22	0	0	1013.90	69.73
343	178042.4	1169828.5	2.7	-17.28	-18.5	15.76	1.22	0	0	1013.90	64.33
343	178042.4	1169828.5	2.7	-18.5	-19.51	16.76	1.01	344	347.44	1361.34	81.23
343	178042.4	1169828.5	2.7	-19.51	-19.54	16.79	0.03	0	0	1361.34	81.08
344	177894.1	1169414.25	6.7	-6.71	-7.62	0.91	0.91	270	245.7	245.70	270.00
344	177894.1	1169414.25	6.7	-7.62	-8.84	2.13	1.22	0	0	245.70	115.35
344	177894.1	1169414.25	6.7	-8.84	-12.5	5.79	3.66	17	62.22	307.92	53.18
344	177894.1	1169414.25	6.7	-12.5	-13.72	7.01	1.22	0	0	307.92	43.93
344	177894.1	1169414.25	6.7	-13.72	-14.94	8.23	1.22	20	24.4	332.32	40.38
344	177894.1	1169414.25	6.7	-14.94	-16.15	9.45	1.22	162	197.64	529.96	56.08
344	177894.1	1169414.25	6.7	-16.15	-17.37	10.67	1.22	71	86.62	616.58	57.79

Where,

DHID: Drill hole identification

XSPM: X-coordinate, m

YSPM: Y-coordinate, m

W_D: Water depth, m

IBEGIN: Sample interval begin, m

IEND: Sample interval end, m

METER: the cumulate thickness of the sample intervals, m

DRIVE: the thickness of each sample interval, m

MGCM: gold grade per meter in composite depth, mg/m^3

G_T: Total gravity in composite depth, mg/m^2

CUMMG: cumulate gold grade of the sample intervals, mg/m^2

CUMMG/CM: average gold grade per meter in depth, mg/m^3.

Table 5.2: Number of samples in the validation dataset for the fish blocks.

Mining Blocks	No of Drill holes (Prediction Dataset)
Coho	36
Halibut	79
Herring	102
Humpy	55
King	71
Pink	56
Red	126
Silver	107
Tomcod	114

5.1 Geostatistical Modeling

5.1.1 Ordinary Kriging for Ore Grade Estimation

Ordinary Kriging (OK) is the most commonly used geostatistical estimation technique. It is often associated with the acronym B.L.U.E known as the "best linear unbiased estimator." Since the kriging estimates are the weighted linear combinations of the available data, it is termed as "linear." It is "unbiased" because of its characteristic property of minimizing the mean error and "best" because of its inherent tendency to minimize the error variance. The theoretical distinguishing feature of OK is its ability to minimize the error variance. However, in practical situations the knowledge of the population error variance and the mean error is not known and the best that can be done is to build a model from the available data and work with the average error and the error variance of the model. Even when the entire available sample data is used, the estimates at the unknown locations will have an associated uncertainty. Therefore in order to reflect the uncertainty, a probabilistic random function model is used in OK. This random function model calculates the bias, the error variance, and the weights of the nearby samples. During the calculation of the OK weights, the random function model requires a pattern of the spatial continuity existing in the data. This is usually obtained from the spatial continuity evident in the sample dataset. The spatial continuity can be defined by fitting a function to the sample variogram.

If the sample values at the known locations are, $V(x_1)$, ..., $V(x_n)$, and suppose one at the unknown location has a true value, $V(x_0)$, then an estimate of this unknown value $\hat{V}$ (x_0) is given by

$$\hat{V}(x_0) = \sum_{i=1}^{n} w_i V(x_i) \qquad (5.1)$$

The weights w_i are obtained from the ordinary kriging system of equations given by

$$\sum_{j=1}^{n} w_j C_{ij} + \mu = C_{i0} \qquad \forall\, i = 1,\ldots,n \qquad (5.2)$$

Where, C_{ij} ---- the covariance

μ ---- Lagrange Parameter

It can be written in matrix notation as

$$\mathbf{C} \cdot \mathbf{W} = \mathbf{D} \qquad (5.3)$$

or

$$\begin{bmatrix} c_{11} \ldots\ldots\ldots\ldots c_{1n} & 1 \\ \ldots\ldots\ldots\ldots\ldots\ldots & \\ c_{n1} \ldots\ldots\ldots\ldots c_{nn} & 1 \\ 1 \ldots\ldots\ldots\ldots 1 & 0 \end{bmatrix}_{(n+1)*(n+1)} \bullet \begin{bmatrix} w_1 \\ \cdot \\ w_n \\ \mu \end{bmatrix}_{(n+1)*1} = \begin{bmatrix} c_{10} \\ \cdot \\ c_{n0} \\ 1 \end{bmatrix}_{(n+1)*1}$$

Multiplying both sides by $\mathbf{C}^{-1}$, the weight matrix is obtained as:

$$\mathbf{W} = \mathbf{C}^{-1} \cdot \mathbf{D} \qquad (5.4)$$

A detailed description on the working principle of the OK and the derivation of the above system of equations is presented elsewhere (Isaaks & Srivastava, 1989).

5.1.2 Nome Ore Grade Estimation Results using OK

The semi-variogram function that describes the spatial continuity for the nine blocks in the Nome deposit was presented in Chapter 4 (Figure 4.13 - Figure 4.21). It is quite evident from all the plots that there is no definite trend in the spatial patterns. The nugget constitutes a major portion of the total sample variability in all the plots. When dealing with geologic datasets, it is common to see a trend or structure. The absence of a trend in the Nome datasets is unusual. This is perhaps due to the sparseness of the data. The mean square error (MSE), mean absolute error (MAE), mean error (ME) and the coefficient of determination (R^2) obtained on the prediction dataset were used as the test statistic to assess the model performance. The MSE criterion which is a measure of the quality of estimation reflects the mean of the square error between the actual and predicted value while the R^2 expressed by equation 5.5 gives an idea of the percentage of the variation captured by the model. As the number of the predictor variables increases the model captures more of the ore grade variation, thereby, resulting in a larger R^2 value. ME or Bias is the average error of estimation and is an indicator for over-estimation or under-estimation. The MAE measures the average absolute deviation between actual and the estimate.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2} \tag{5.5}$$

Where y_i is the true ore grade, $\bar{y}$ is the mean ore grade and $\hat{y}$ is the predicted ore grade value. Apart from the R^2 being used as one of the yardstick for evaluating the model performance, there are a number of other statistics that can be used for evaluating the effectiveness of a model. These measures such as the confusion matrix, kappa statistic and the receiver operating characteristic (ROC) are, however, applicable towards the classification tasks. The confusion matrix is a sort of classification table of the predicted and observed values based on the fitted model. The measure of accuracy obtained from this matrix is sometimes misleading owing to the fact that its interpretation depends on knowledge of the prior probability of occurrence of the subject in question (Boyce et al., 2002). Apart from this, in the confusion matrix a bias is also introduced when a continuous variable is dichotomized. This limitation is overcome by ROC which is independent of the probability cut-off levels.

Generally, a model with less error and high R^2 is considered as the one generating a better fit. The estimates obtained on the validation dataset for all the nine fish blocks are shown in Table 5.3. The results are expectedly poor because ordinary kriging produces local linear estimates. Therefore, attempt has been made to model the deposit using non-linear artificial machine learning algorithms. These are presented in sections 5.2 and 5.3 in this chapter. Similar studies of applying the kriging technique for building the model and using it for predicting on a validation dataset has also been reported in spatial interpolation comparison (SIC-2004) (Dubois, 2005).

Table 5.3 OK estimates for the fish blocks.

Data Set	Mean Error	Mean Absolute Error	Mean Squared Error	R^2 (in %)
Coho	-91.50	729.20	1855934.54	0.07
Halibut	2.55	207.94	132351.88	11.10
Herring	-5.83	566.98	268719.57	8.80
Humpy	75.76	484.27	446436.34	0.40
King	8.17	329.44	298962.30	30.50
Pink	33.83	128.97	47171.00	3.00
Red	33.54	353.02	319487.8	19.30
Silver	12.58	221.46	138499.46	7.70
Tomcod	-6.02	15.037	141441.00	17.50

5.2 Neural Network Modeling

5.2.1 Neural Network for Ore Grade Estimation

Neural network (NN) is a computational structure inspired by the study of biological neural processing. This structure exhibits certain brain-like capabilities, including perception, pattern recognition, and pattern prediction in a variety of situations. As in the brain, information processing is done in parallel using a network of "neurons." As a result, NNs have capabilities that go beyond algorithmic programming and work exceptionally well for non-linear input-output mapping. It is this property of non-linear mapping that makes neural network appealing for ore grade estimation.

There is a fundamental difference in the principles of OK and NN. While OK utilizes information from local samples only, NN utilizes information from all of the samples. OK is regarded as a local estimation technique, whereas NN is a global estimation technique. If any non-linear spatial trend is present in a deposit, it is expected that the neural network will capture it reasonably well. The basic mechanisms of neural networks have been discussed at length in literature (Bishop, 1995; Hagan et al., 1995). A brief discussion on NN theory is presented below to provide an overview of the topic.

Neural networks process information in interconnected layers. A layer is simply a group of elements designated as neurons. There is the input layer consisting of the inputs, one or more *hidden layers* consisting of a number of neurons, and the output layer consisting of the outputs. Thus, a neural network has three or more layers. Typical network architecture is presented in Figure 5.1. The network present in the figure has 3 layers. The inputs (elements in the input layer) are first manipulated to determine the value of the elements of the hidden layer. It should be noted that while the input layer and the output layer have a fixed number of elements for a given problem, the number of elements in the hidden layer is arbitrary.

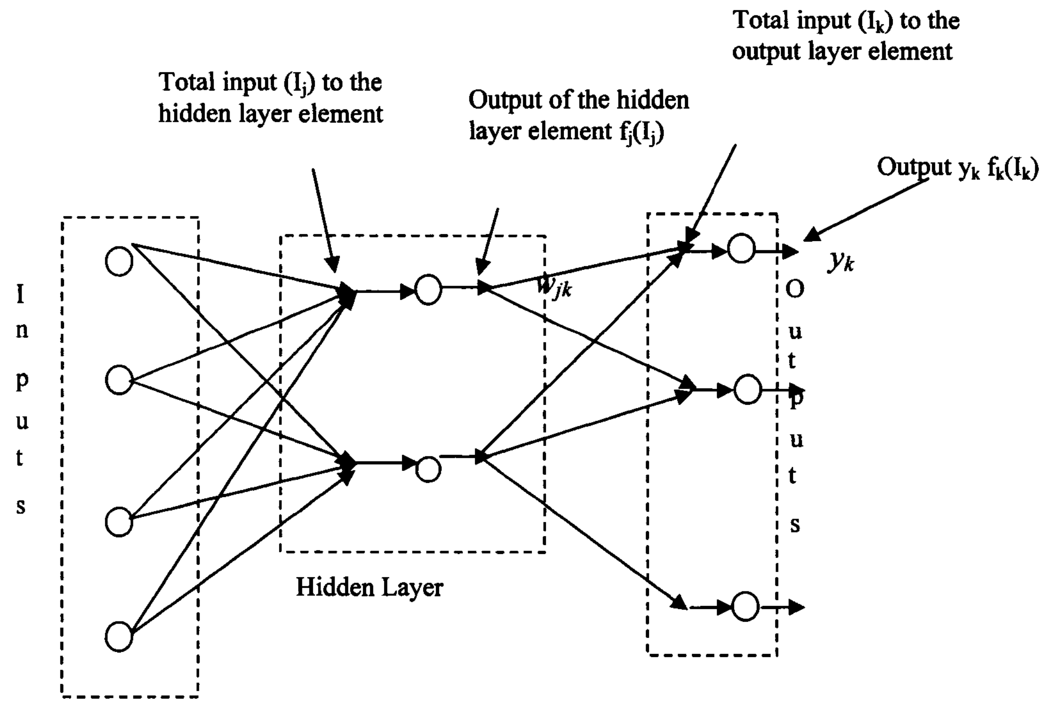

Total input (I_k) to the output layer element
Total input (I_j) to the hidden layer element
Output of the hidden layer element f_j(I_j)
Output y_k f_k(I_k)
y_k
w_jk
Inputs
Hidden Layer
Outputs

Figure 5.1: A neural network architecture

Each input x_i is assigned a weight w_{ij} and connected to element j of the hidden layer. Each input is connected to each node of the hidden layer just as each node of the hidden layer is connected to each node of the output layer. Mathematically, each hidden layer element is equal to the sum of the product of the inputs and their assigned weights. The output of each hidden layer element is obtained by applying a function (f_j) to the sum (i.e. the input of the element). As in the case of the input layer, this output, in turn, is weighted (w_{jk}) and sent as inputs to each of the output elements. The final output, i.e. each output element, y_k, is obtained by applying a function (f_k) to the sum of the inputs from the hidden layer. Therefore,

$$y_k = f_k\left(\sum_{i\to j} w_{jk} f_j\left(\sum_{i\to j} w_{ij} x_i\right)\right) \tag{5.6}$$

In the present analysis, as there is only one output, the generic output will be referred to as y rather than y_k. The functions $f_j(\bullet)$ and $f_k(\bullet)$ are typically a Logistic function as shown below:

$$f_x = \frac{1}{1+e^{-x}} \tag{5.7}$$

However, other activation functions like tanh, gaussian, gaussian-complementary or their combinations might be used.

From the preceding discussion, it may be apparent that the obtained output depends on the weights of the interconnections. Therefore, it is obvious that these weights need to be determined in such a way that produces accurate outputs. The process of determination of weights is called "learning" or "training." Learning is a process by which the neural network adjusts its weights to reflect the changes in the data fluctuations in a spatial coordinate. During learning, the neural network maps the patterns pre-existing in the data. The sample data set for a given deposit is used for this purpose. Assume that there are n sample points. The network starts learning with sample point number one. It begins with arbitrary weights; typically, all weights are set to random numbers with a mean of zero and a standard deviation of one. In most cases, it doesn't matter what initial weights are chosen. Using the initial weights, the network computes the output y as described previously. Depending on the output, weights are adjusted based on their contribution to the error. This process of propagating the effect of the error onto all the weights is called *back propagation.* Using the adjusted weights, the network then computes an output for sample number two. The weights are adjusted once again to reflect the error. Ultimately, it goes through all the samples, adjusting the weights on each occasion. When a neural network is done training on a sample set, it is said to have completed an *epoch*. The process of completing epochs is called online learning. There is another method called batch learning where the error is calculated on an individual sample basis. In this method, the weight is updated only once in an epoch, with the new weight being based on the errors of all the samples in the training data set.

Typically, the weights after a single epoch are usually unsatisfactory. To make the network satisfactory, the learning/training process is restarted with the initial weights being the final weights of the previous epoch. When the error for an epoch is deemed satisfactory, the network is considered "trained."

5.2.2 Optimization of Neural Network Training

Among the many practical modeling issues, neural network training is still debated. A neural network model working on a grade estimation problem performs mappings from an input space to an output space. For example, given the spatial co-ordinates and other relevant attributes as input, and grade attribute as output, neural network will be able to generate a mapping function through a set of connection weights between the input and the output. Hence, output O of a neural network can be regarded as a function of inputs X and connection weights W: $O = \varphi (X)$, where φ is a mapping function. Training of a neural network involves finding a good mapping function that maps the input-output patterns correctly. This is done, as previously described, by adjusting the connection weights between the neurons of a network using a suitable learning algorithm, while simultaneously fixing the network architecture and activation function. In essence, given a set of training patterns consisting of input-output data pairs of spatial co-ordinates and other attributes and grade attribute $\{(I_1, D_1), (I_2, D_2) \ldots (I_n, D_n)$, the learning algorithm strives to minimize the training error. One popular error function is a squared-error function in which error is described as, $e (W, I_i, D_i) = (\varphi (I_i, W) - D_i)^2$. Using a suitable learning rule, a set of connection weights, W, is found so that the squared error function is minimized.

In a multilayer feed forward neural network, a supervised learning algorithm is applied to train a network. Supervised learning used in neural network training can be considered as an unconstrained nonlinear optimization problem in which the objective function, defined here as the squared error function, should be minimized in the search of weight space. Error functions spanned by the weight space in a neural network may have a single minimum as the global minimum. On the other hand, an error function may generate a very complicated error surface with many local minima in the weight space, where only one is a global minimum. Gallaher (1999) argued that a local minimum in neural network is not a major problem, where as Shang and Wang (1996) showed that the error surface could be very rugged and may have several local minima. For the first case, local learning algorithms may be adequate. Obviously in the presence of many local minima, local learning algorithms could have difficulty in finding the optimal solution and may get trapped in the local minimum point. In the present context, an investigation was carried out to observe the performance of few local learning algorithms for neural network modeling and was applied for ore grade estimation in the nome area.

Three local optimization techniques that were investigated are: (i) standard gradient descent back-propagation (SBP) with fixed learning rate, (ii) back-propagation with momentum learning (MBP), and (iii) Levenberg-Marquardt back-propagation (LMBP) learning. Comparative evaluation of these techniques in neural network learning optimization was also carried out with the data sets from Nome area.

5.2.2.1 Local Learning Algorithms

Learning the weights of a neural network can be considered to be an unconstrained continuous non-linear minimization problem. In the past, many techniques have been developed for solving non-linear optimization problems (Battiti, 1992; Dixon, 1994). Optimization methods can be classified into local optimization and global optimization. Examples of local optimization techniques include gradient descent algorithm, Newton's method, and the conjugate-gradient method. These techniques are also applicable in neural network learning as well.

Each of the local optimization techniques use some form of gradient information in their search strategy, and requires the calculation of the gradient of error with respect to the weight vector. Due to hidden layers topology in a neural network, it is not possible for a direct calculation of the error gradient with respect to the hidden layers connection weights. Instead, an algorithm called back-propagation is used to calculate the gradient. Back-propagation algorithms apply a chain rule for calculating the gradient, which is done by back propagating the sensitivities (change of error function with respect to net input to a neuron) from output layers to previous layers step by step in the backward direction (output-hidden-input). This study applied three local optimization algorithms, the properties of which were studied in the Nome ore reserve estimation problem. The basic mechanisms and mathematical foundation of these techniques can be found in published literature (Haykins, 1998; Bishop, 1995; Hagan, 1995). For completeness a brief overview of the optimization is presented below.

5.2.2.1.1 Standard back propagation with gradient descent (SBP)

The gradient-descent algorithm finds a locally optimal solution by iteratively taking small steps in the gradient descent direction. The search procedure starts with a random initial guess of parameters in the weight space. The weight is updated in each iteration according to following equation:

$$w\,(n+1) = w(n) - \eta\,\nabla e(w) \tag{5.8}$$

Where, η is learning rate parameter.

$\nabla e(w)$ is the gradient of error.

The learning rate parameter, η, plays a major role in convergence of the algorithm. A small learning parameter causes a small change of the weight vector along the gradient descent direction, which results in very slow progress along the search trajectory. On the other hand, a large learning parameter may cause an overshoot of the minimum point, although it provides a faster convergence. Back-propagation is the generalization of this gradient-descent algorithm, which employs the chain rule in calculating gradients at hidden layers. A complete description of back propagation algorithm can be found in Hagan (1995).

5.2.2.1.2 Back propagation with momentum (MBP)

Use of a large learning rate causes gradient descent algorithms to oscillate along the search trajectory, and may even cause divergence. In order to get the full benefit of a faster convergence with large learning rate, the oscillation along the search path must be reduced. Use of a momentum algorithm facilitates the dampening of the oscillations and renders fast convergence. Momentum algorithm introduces a momentum factor and makes the new weight change as follows:

$$\Delta w(n) = \gamma\, \Delta w(n-1) - (1-\gamma).\eta\, \nabla e(w) \quad (5.9)$$

Where, $\Delta w(n) = w(n) - w(n-1)$

γ = momentum co-efficient, $0 \leq \gamma < 1$

$\nabla e(w)$ = the gradient of error

5.2.2.1.3 Levenberg-Marquardt algorithms (LMBP)

The Levenberg-Marquardt algorithm is a modification of the Newton's method for non-linear optimization. The Levenberg-Marquardt and Newton method use, in addition to the gradient, other numerical quantities such as the Hessian matrix of the error surface, which consists of the 2nd order derivative of the error function. These methods are also based on the concept of quadratic approximation of the error function in a local region. If the error function is truly quadratic in nature, the Newton's method finds the minimum solution in a single iteration. Therefore, the success of this technique depends upon how closely the error function resembles the quadratic function. Even the

Levenberg-Marquardt algorithm will diverge if the quadratic approximation is not appropriate. Searching for an optimal solution using this method requires the calculation of the inverse of the Hessian matrix, which should be positive definite. Newton's method does not always guarantee the positive definiteness of Hessian matrix. The Levenberg-Marquardt introduces a regularization term into the Hessian matrix so that positive definiteness of the Hessian matrix is guaranteed.

5.2.2.2 Model Generalization

5.2.2.2.1 Quick-Stop Training

A simple criterion for optimization of the neural network architecture is to choose the network with minimal generalization error. The quality of the predictions made by the network is measured in terms of the so-called generalization error. The main goal of neural network modeling is not to exactly fit to the training data, but to generalize a model which well-represents the underlying characteristic of a process. A simple model may result in poor generalization, since it cannot take into account all the intricacies that may be present in the data. On the other hand, too complex a model is flexible enough to fit data with anomalies or noise. Therefore, complexity of a model should be well matched to improve generalization properties of the data. Much research was devoted in the past to improve the generalization of NN models, with results that included techniques such as regularization, quick-stop training, smoothing and training with noise, and combining several learning models using various ensemble techniques like Bragging and Boosting. In this research a quick-stop training method is employed to improve the NN model generalization.

The quick stop-training works under the notion that generalization performance varies over time as the network adapts during training (Hagan et al., 1995). Randomly selected initial weights are likely to be completely inconsistent with the samples, so both the training set and the generalization errors are likely to be high before learning begins.

During training, the network adapts to decrease the error on the training patterns. In the early stages of learning, the generalization error tends to decrease in step with the training error as the network captures the major features of the underlying function. If the training data is noisy or incomplete, there may still be a possibility that a trend may exist. In such situation, in addition to representing the general properties of the target function, it is likely to contain peculiarities unique to the particular data set and uncharacteristic of the target function. As these idiosyncrasies are exploited in later stages of learning, the improvement in generalization that comes from being right on the training examples is offset by errors introduced elsewhere and the generalization error begins to increase again even though the training error continues to decrease.

By using the quick stop training method, the data set is split into three subsets, training, calibration and validation. The network actually undergoes training on the training set. However, the decision to stop the training is made on the network's performance in the calibration set. The training error for the training set decreases monotonically for an increasing number of epochs, in the usual manner. In contrast, the error for the calibration set decreases monotonically to a minimum, and then starts to increase as the training continues. A typical profile of training and calibration error of neural network model looks like a pattern as presented in Figure 5.2. This observed behavior is due to the fact that, unlike the training data, the calibration error is not used to train the network. The calibration data is simply used as an independent measure of the model performance. Therefore, it is possible to stop over-training by monitoring the performance of the network on the calibration subset, and then stopping the training when the calibration error starts increasing.

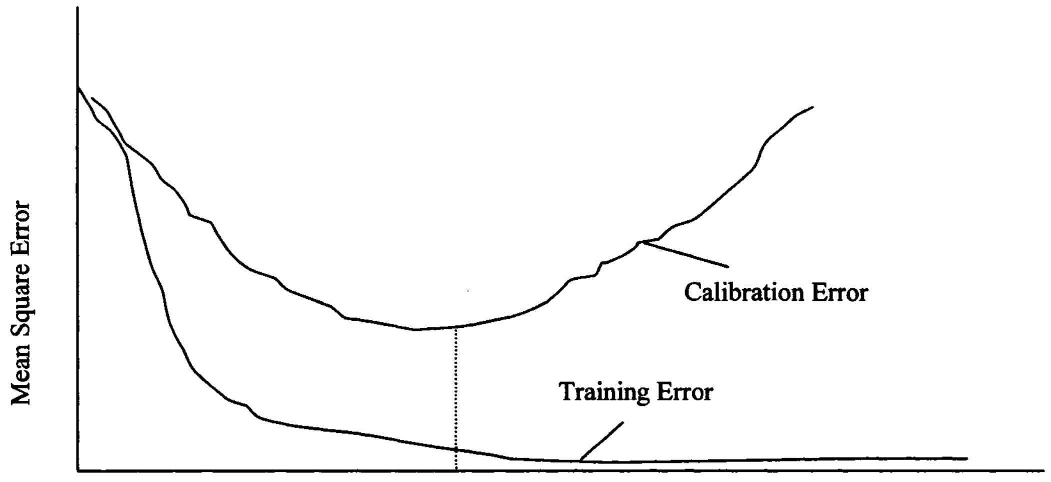


Figure 5.2: A typical profile of training and calibration error of NN Model (Haykins, 1999)

5.2.2.3 Model Performance Measurement

A direct way to estimate the generalization ability of a system is to measure the error on a separate data set that the network has not seen during training. The neural network model performance can be improved in terms of generalization error, if a suitable validation of the model is carried out. Since the model generalization error is evaluated on a validation data set other than the training set, the entire data set might be divided into two sets: the training set and the validation set (for simple validation of the model). Since quick-stop training is also used to improve model generalization, the training data set can again be divided to obtain a calibration data set. Therefore, the entire data set is divided into three subsets: training, calibration and validation.

In order to make a valid model performance measurement, training, calibration and validation subsets should, however, have similar statistical properties. Therefore, the members of the data in the training, calibration and the validation subsets should be selected in such a way that the three data subsets acquire similar statistical properties. The traditional approach for the data division employs a random division of data into training, calibration and validation subsets. Random selection of data may not be appropriate. Therefore, an integrated approach using data segmentation and genetic algorithm as described in the previous chapter was applied for appropriate division of the data.

5.2.3 NN Nome Ore Grade Estimation Results

Separate neural networks models were developed for each of the fish blocks. The architecture selected for the NN modeling was a ward-net network. An advantage of ward-net architecture is that this type of network employs different activation functions in the hidden layers. As a result, complex non-linear input-output pattern is captured by a combination of multiple hidden units with different activation functions. Although the same network architecture was used for the entire datasets, the number of hidden units was varied for each individual fish block. Mixture of Logistic, Tanh, Gaussian, and Gaussian Complimentary activation functions were used in the network. A general configuration of the ward-net architecture used in this study is presented in Figure 6.4. The architecture has three layers: the input layer, the hidden layer, and the output layer. The hidden layer consists of three hidden slabs of Gaussian, Tanh, and a second Gaussian activation functions. The number of input and output nodes is dictated by the problem. In this case, three nodes for the three input variables: X co-ordinate, Y co-ordinate, and Water-table depth, and one node for a single output variable of gold attribute. The number of nodes in the hidden slabs varies for different data sets. The number of hidden neurons used for each of the fish blocks is presented in Table 5.4. The number of hidden neurons was chosen based on the minimum generalization errors of the NN models while experimenting with a different number of hidden nodes in the hidden slabs. MATLAB code was developed for conducting all the studies associated with NN modeling.

5.2.3.1 Performance of optimization algorithms

Prior to applying the neural network model, the output data values were normalized. Therefore, the error presented is based on the normalized transformed data. The neural network model was trained using the local learning algorithms. The behavior of the local learning algorithms was examined for all the data sets mentioned earlier using the three local learning algorithms. The starting point for all the learning algorithms, local and global, was the same. As an example, the properties of the learning algorithms for the king data set are presented in Figure 5.3.

Figure 5.3(a) shows the learning curve of the three algorithms for the King data set with learning rate ($\eta = 0.4$), and momentum ($\gamma = 0.4$). The mean squared error for the LMBP algorithm decreases rapidly with the increased learning epochs. Hence, the LMBP algorithm provides a faster convergence than the other two algorithms. The SBP algorithm learns very slowly since the learning rate parameter is small. The MBP algorithm is even slower when compared to SBP. However, SBP and MBP algorithms are faster when the learning rate was increased to 2. In spite of the increased learning rate the time to converge still lags the LMBP (Figure 5.3 (b)). After learning the pattern of the local algorithms for first few epochs, the model was run for 20,000 epochs. The best solution, in terms of minimum mean squared error, within 20,000 epochs was then obtained for the three algorithms. Table 5.5 presents the best solutions for the three algorithms. From the table, it can be observed that the LMBP algorithm provided superior performance for all the data sets. The minimum mean squared errors were the least for the LMBP algorithm.

Table 5.4: Number of neurons for various data sets in NN modeling

Data Set	No. of Input	No. of Hidden	No. of output
Coho	3	24	1
Halibut	3	30	1
Herring	3	30	1
Humpy	3	6	1
King	3	15	1
Pink	3	36	1
Red	3	12	1
Silver	3	9	1
Tomcod	3	15	1

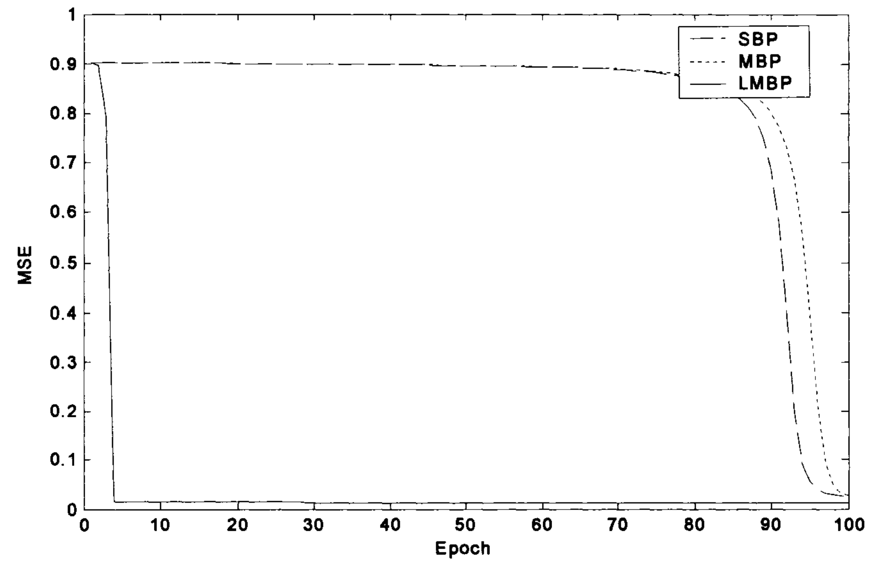


Figure 5.3 (a): Network learning with various learning algorithms (King data) (γ=.4, α=.4)

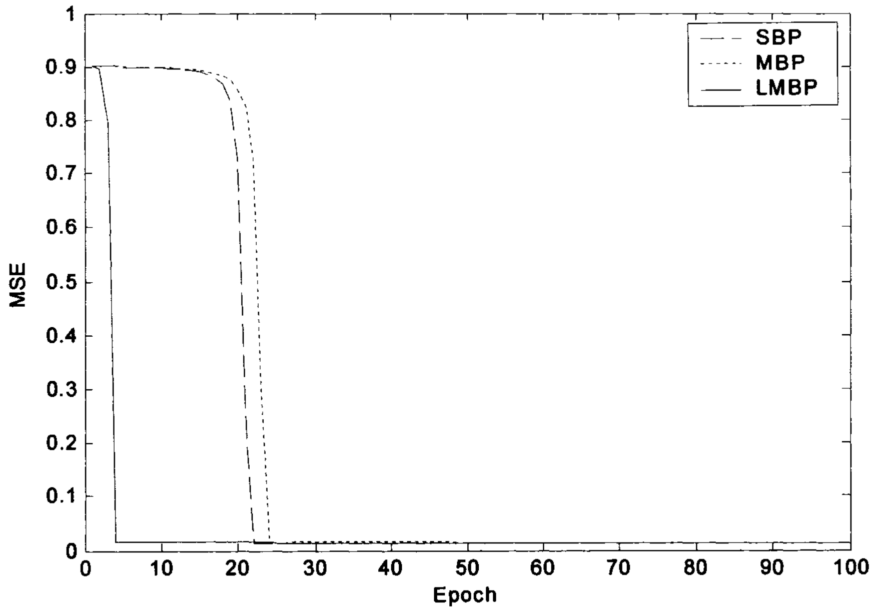


Figure 5.3(b): Network learning with various learning algorithms (King data) (γ=.4, α=2)

Table 5.5: Performance of various local learning algorithms

Data set	Learning algorithm	Minimum MSE
Coho	SBP	.0176
	MBP	.0176
	LMBP	.0043
Halibut	SBP	.0281
	MBP	.0147
	LMBP	.0070
Herring	SBP	.0065
	MBP	.0087
	LMBP	.0028
Humpy	SBP	.0163
	MBP	.0166
	LMBP	.0083
King	SBP	.0099
	MBP	.0101
	LMBP	.0032
Pink	SBP	.0168
	MBP	.0168
	LMBP	.0030
Red	SBP	.0176
	MBP	.0176
	LMBP	.0043
Silver	SBP	.0193
	MBP	.0194
	LMBP	.0073
Tomcod	SBP	.0263
	MBP	.0263
	LMBP	.0100

The results presented above pertain to the neural network performance on the training data. They indicate the comparative performance of various learning algorithms while the neural network is on training. A minimum training error does not guarantee a reliable estimate of the predictive ability of the model. Since, the model was run for large number of epochs; it was not known a priori whether the network was over-trained. An in-depth analysis indicated that the neural network model was over-trained. As an example, overtraining could be seen from the performance of the neural network model using LMBP algorithm on a training and associated validation data sets for the King block (Figure 5.4 (a & b)). From the figures, it is seen that the neural network model gets is excessively overfitted with the training data. The R^2 values for the training and the validation data sets are 0.79 and 0.12 respectively. Hence, the trained model overstates the model performance in the validation dataset.

Since the predictive quality can only be judged based on the generalization performance of a model, an attempt was made to improve the generalization ability of the neural network model using the quick-stop training method. In this method, the network was trained using the training data, and its performance was observed on the calibration data on an epoch by epoch basis. The model was selected when the minimum mean squared error is reached on the calibration data. However, the training of the model was continued for another 10,000 epochs after reaching the minimum error to ensure the stability of the solution. It was also verified whether the solution is stable to 20,000 epochs.

Table 5.6 presents summary statistics of the generalization performance of the neural network model for all the fish blocks. The number of data points in the validation dataset for each block is presented in Table 5.2. Furthermore, the scatter plots of the actual vs. the predicted values for the validation data are also presented in Figures 5.5 through 5.13. From these results, it can be seen that the neural network model performance is not adequate for the Nome gold data sets. Both the mean squared error and R^2 values indicate poor model generalization.

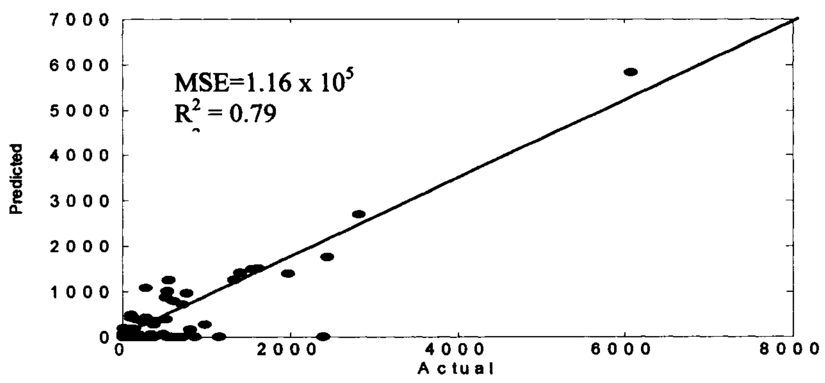


Figure 5.4 (a): Actual vs. predicted for the training data on King data set

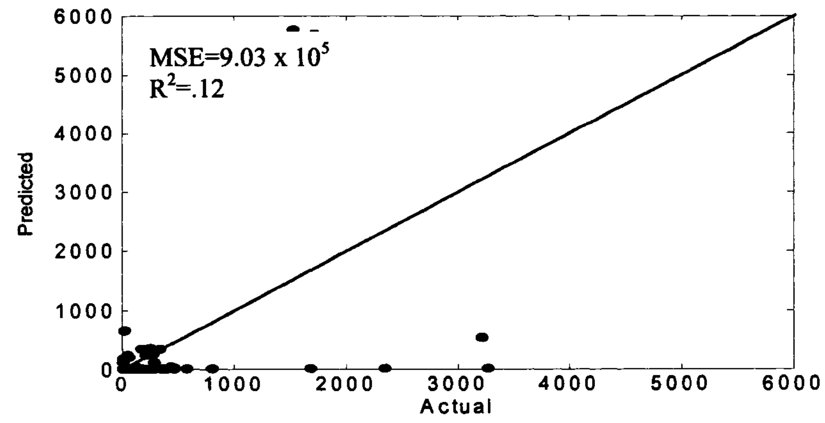


Figure 5.4 (b): Actual vs. predicted for the validation data on King data set

Table 5.6: Generalization performance of NN models in fish blocks

Data Set	Bias	Mean Absolute Error	Mean Squared Error	R^2 (in %)
Coho	-33.44	704.01	1294300.00	12.20
Halibut	-3.59	230.00	153650.00	3.10
Herring	-37.25	304.70	259850.00	10.20
Humpy	-44.29	453.79	444090.00	2.10
King	25.43	346.79	351240.00	19.30
Pink	-13.03	108.66	24576.00	5.10
Red	-1.30	351.50	318490.00	19.10
Silver	-16.42	226.30	143540.00	3.20
Tomcod	0.87	222.66	152150.00	10.50

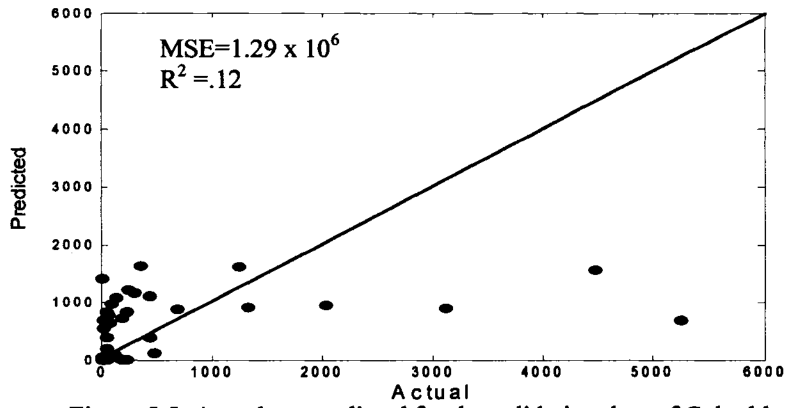


Figure 5.5: Actual vs. predicted for the validation data of Coho block

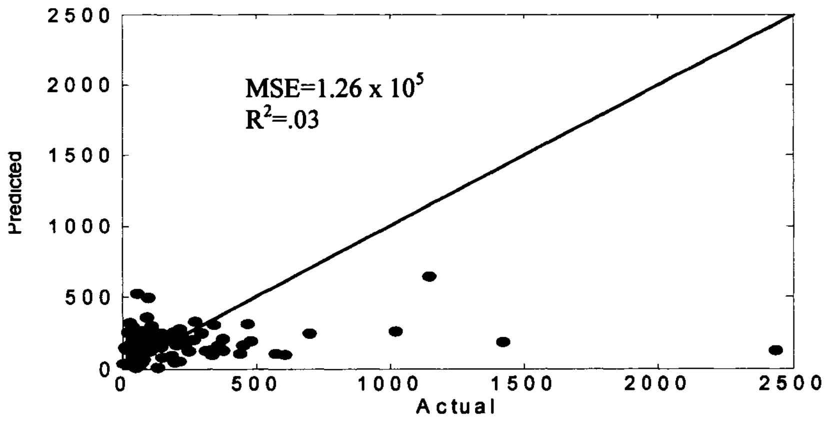


Figure 5.6: Actual vs. predicted for the validation data of Halibut block

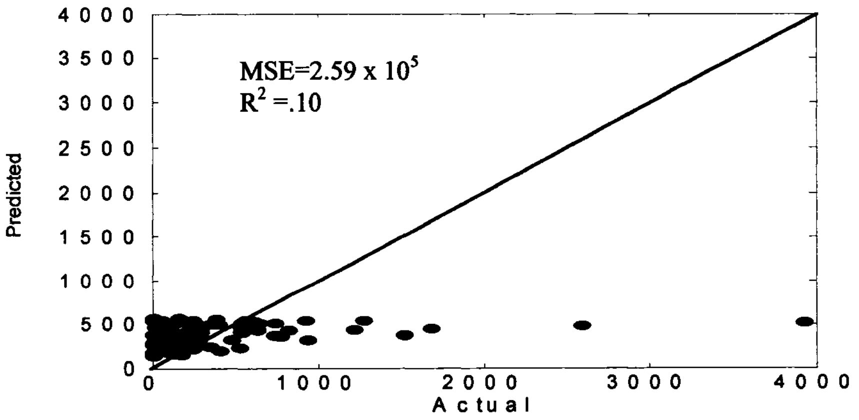


Figure 5.7: Actual vs. predicted for the validation data of Herring block

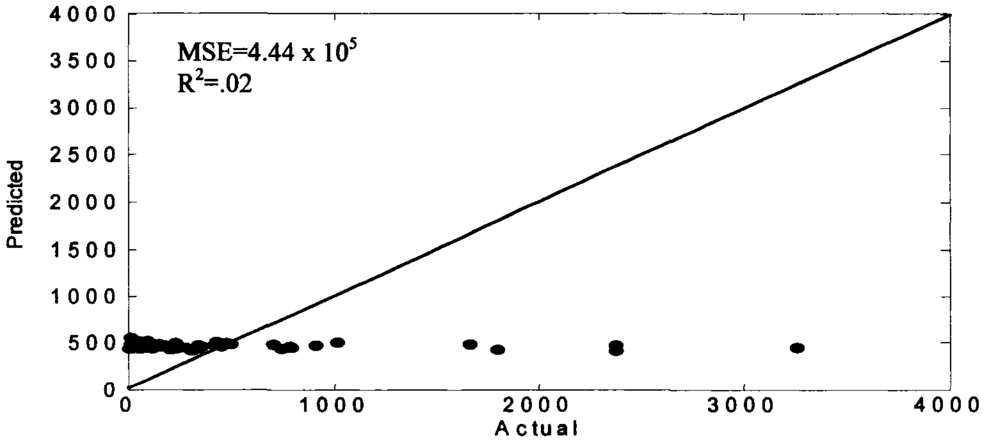


Figure 5.8: Actual vs. predicted for the validation data of Humpy block

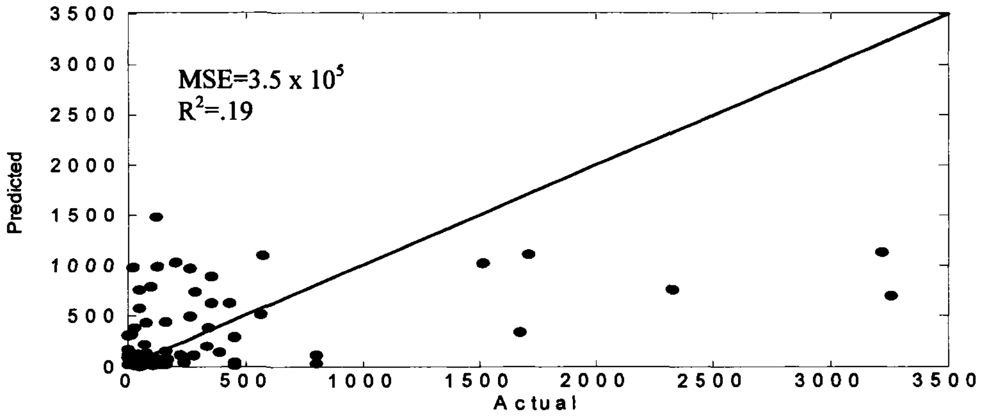


Figure 5.9: Actual vs. predicted for the validation data of King block

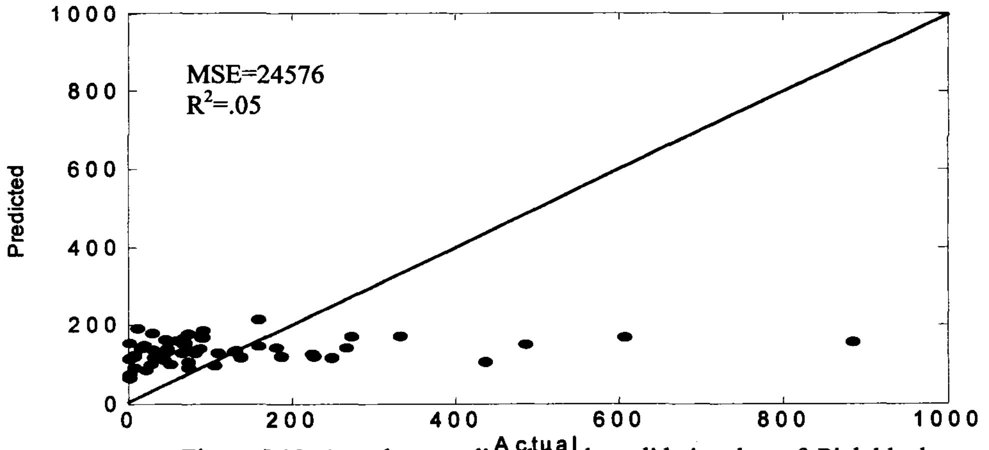


Figure 5.10: Actual vs. predicted for the validation data of Pink block

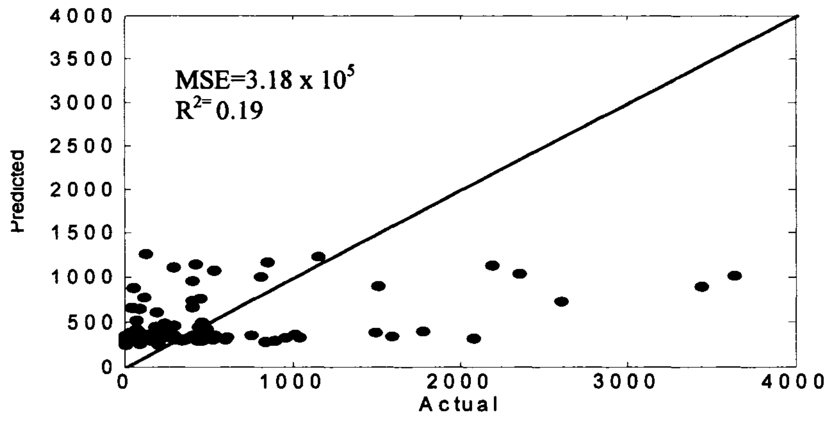


Figure 5.11: Actual vs. predicted for the validation data of Red block

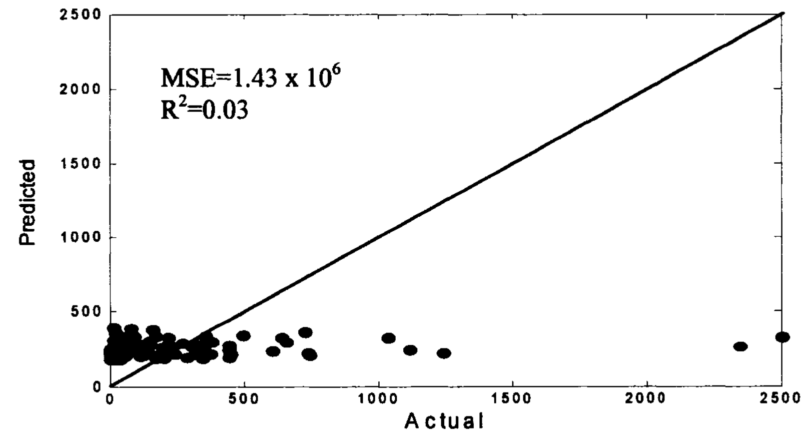


Figure 5.12: Actual vs. predicted for the validation data of Silver block

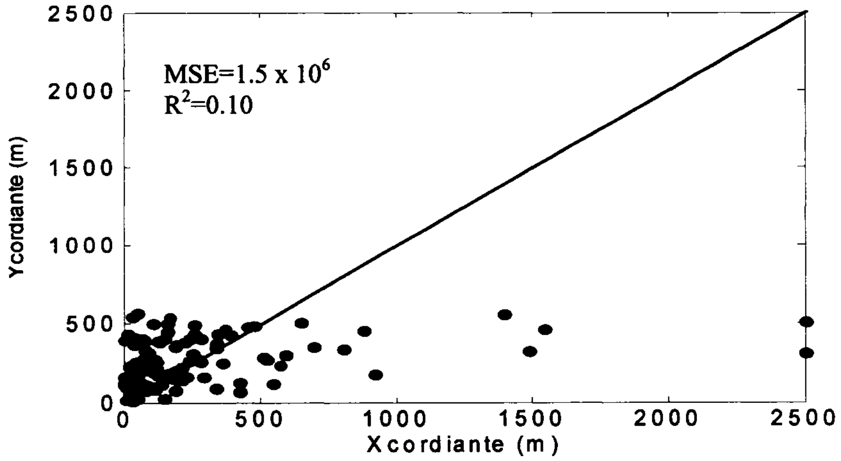


Figure 5.13: Actual vs. predicted for the validation data of Tomcod block

5.3 Support Vector Machine Modeling

5.3.1 SVM for Ore Grade Estimation

The regression abilities of the support vector machine can be used for grade estimation purposes. During this process, a mapping function is developed between the input variables (spatial coordinates, secondary variables) and the output variables (ore grades). The principle of SVR approach is described previously in chapter 3. Broadly, the methodology for the ore grade estimation consists of three distinct stages: (a) transformation and preprocessing of data, (b) model application, (c) post-processing of the model results. For this particular application, the placer gold grade is related to the spatial coordinates and the water table depth by a support-vector machine based non-linear model. Stage (a) involves preparing the raw data for model simulation. Ke (2002) explained in detail the steps involved in the transformation and the preprocessing of the data. The reserve estimation was performed separately for each of the nine blocks. Stage (b) involves the model application to the input data. The recommendations of Kecman (2004), Change and Lin (2001) were considered carefully in the development of the SVM-based model. As per the recommendations, the input data is first scaled. The scaling of the independent variables (spatial coordinates, water table depth) and the output variable (ore grade) was done assuming a uniform distribution. In other words, the scaled value of an attribute is calculated using the maximum and the minimum values of the attribute. The next step in developing an SVM-model is to select a kernel for linearizing the problem.

Many options are available for kernel selection in SVM. Table 5.7 presents a list of the commonly used kernels. For this study, a radial basis kernel (RBF) is selected as it is translation invariant and the parameters are estimated during the cross-validation stage.

Table 5.7 Commonly used kernels in SVM.

Kernel Type	Expression
Simple dot product[a]	$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}*\mathbf{x}'$
Polynomial	$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}*\mathbf{x}' + 1)^d$, d is user specified
Two-layer neural network	$K(\mathbf{x}, \mathbf{x}') = \tanh(b(\mathbf{x}*\mathbf{x}') - c))$, b and c are user specified
Radial basis[b]	$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma^2 \|\mathbf{x} - \mathbf{x}'\|^2)$, γ^2 is user specified

[a]This kernel corresponds to linear machine.

[b]This kernel is translation invariant. Can be written as Gaussian covariance kernel with unit variance: $K(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{r^2}\right) = \exp\left(-\frac{h^2}{r^2}\right)$, where $\sigma^2 = 1$, $r^2 = 1/\gamma^2$, and $h^2 = \|\mathbf{x} - \mathbf{x}'\|^2$.

The drill holes for each block (Table 4.1) were used to estimate the SVM parameters, the cost function (C), the radial basis kernel parameter (γ) and the error margin (€). One way of conducting the training/ validation is with a split sample approach. This approach divides the available data into two subsets and uses one for the training and the remaining for the validation. Optimal SVM parameters were determined based on a K-fold cross-validation technique applied to the training dataset. The K-fold cross-validation approach splits the available data into more or less K equal parts. Of the K parts of the data, only K-1 parts of the data were used to find the SVM estimate and calculate the error of the fitted model, and for predicting the k^{th} part of the data as part of the validation process. The procedure then continues for k = 1, 2, . . ., K, and the selection of parameters is based on the minimum prediction error estimates over all K parts. The selection of K based on the shape of a "learning curve" (Hastie et al. (2001)). A learning curve is a plot of the training error versus the training size. For given SVM parameters (γ, C and €), different training errors are calculated by progressively estimating the SVM model for increased number of the training size, constituting a plot of the learning curve. From the learning curve an optimum training size (or in other words the number of folds, K) can be obtained where the error is minimal.

Even though the actual value of the training error may differ for different combinations of SVM parameters, the shape of the learning curve remains more or less the same (i.e., the training size that corresponds to the optimum point). The learning curve procedure was used for each of the blocks in the Nome area to train and to validate the model. Once the optimal number of folds, K, is selected, the SVM model is trained using the K-fold cross validation. The training and testing involves a thorough grid-search for the optimal C and γ values.

Stage (c) involves post-processing of the model results during which maps are developed to show the gold grade distribution in each block. The predicted gold grades in an individual block were used to estimate the maximum mineral inventory within 5 meters of the sediment surface. Cutoff grades of 300, 400, 500, 600 and 700 mg/cu.m were used for calculation of the in-place mineral reserve.

5.3.2 SVM Nome Ore Grade Estimation Results

In this study the SVM based model was trained and validated using a ten -fold cross-validation. Unlike the neural networks where training involves passing a data set through hidden layers to optimize the weights, optimal training of SVM involves estimation of the parameters, C and γ through a grid search such that the error is minimized.

Table 5.8. Cost function and kernel width values for the blocks

	Coho	*Halibut*	*Herring*	*Humpy*	*King*	*Pink*	*Red*	*Silver*	*Tomcod*
C	2	2	1.2	0.6	0.8	1.4	0.53	0.5	1.8
γ	2	2	2.5	4.5	2.3	3.5	9.5	3.1	5.5

The optimal C and γ values (Table 5.8) for each of the blocks were selected from different combinations of C and γ values depicted by the troughs and the flat regions of the error surfaces (Figure 5.14- Figure 5.22). Once the optimal values for the SVM model parameters were obtained the model was tested for its generalization ability on the validation datasets for each of the fish blocks shown in Table 5.2. Figure 5.23-Figure 5.31 show the performance of the SVM model in predicting the gold grade for the nine blocks. The performance statistics of the model are presented in Table 5.9. It could be seen from the Table 5.9 that the SVM based model produced better estimates compared to the other two methods. However, the improvement was only marginal which may be due to the presence of the extreme data values.

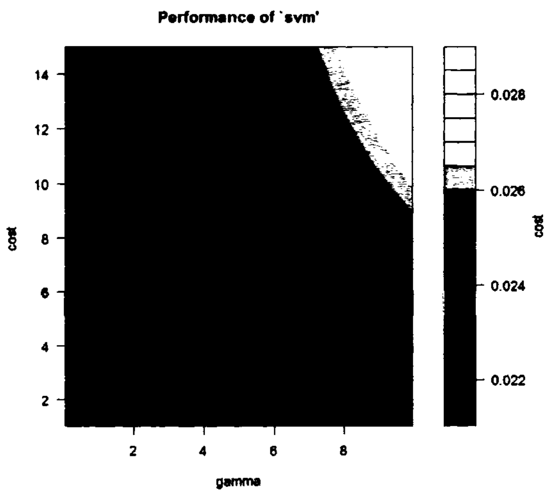


Figure 5.14: Effect of cost and kernel width variation on error (Pink)

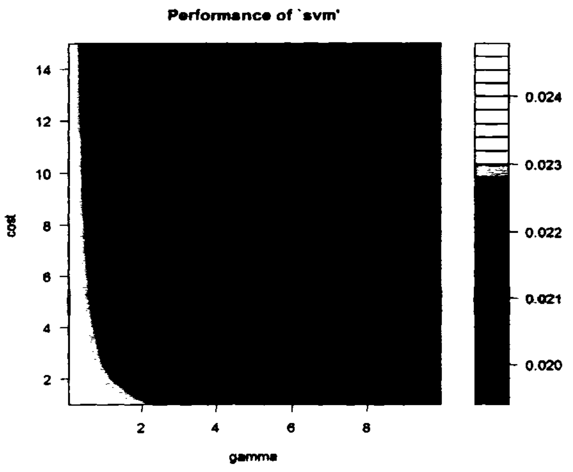


Figure 5.15: Effect of cost and kernel width variation on error (Red)

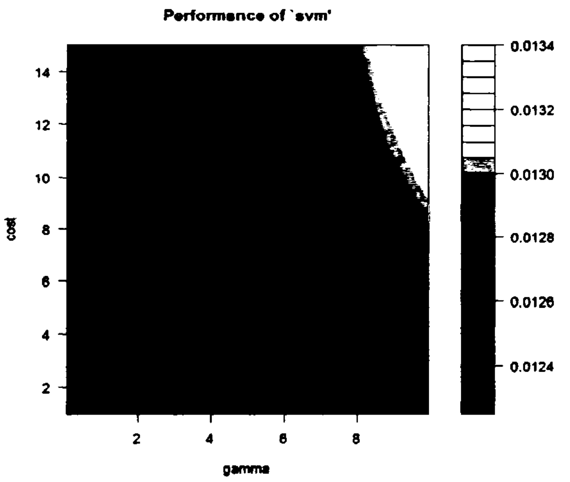


Figure 5.16: Effect of cost and kernel width variation on error (Halibut)

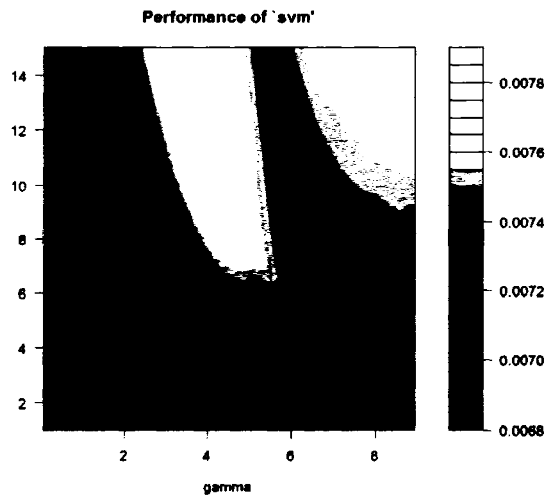


Figure 5.17: Effect of cost and kernel width variation on error (Herring)

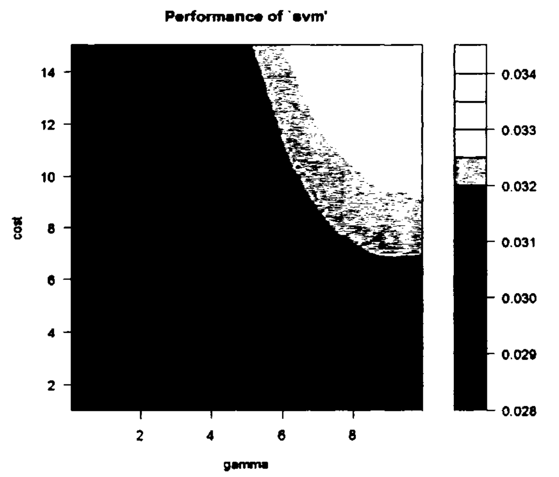


Figure 5.18: Effect of cost and kernel width variation on error (Humpy)

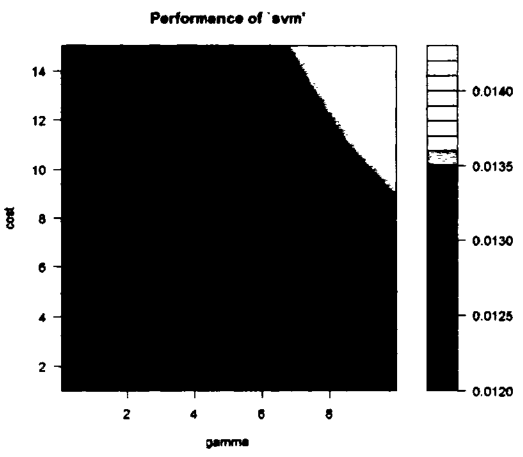


Figure 5.19: Effect of cost and kernel width variation on error (King)

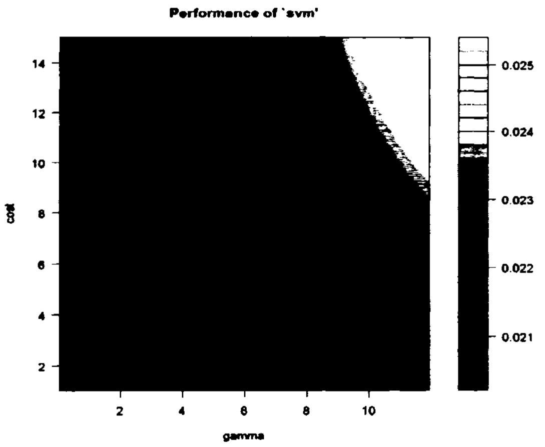


Figure 5.20: Effect of cost and kernel width variation on error (Silver)

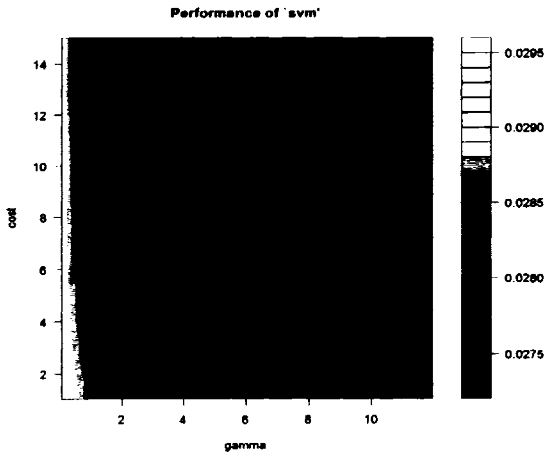


Figure 5.21: Effect of cost and kernel width variation on error (Tomcod)

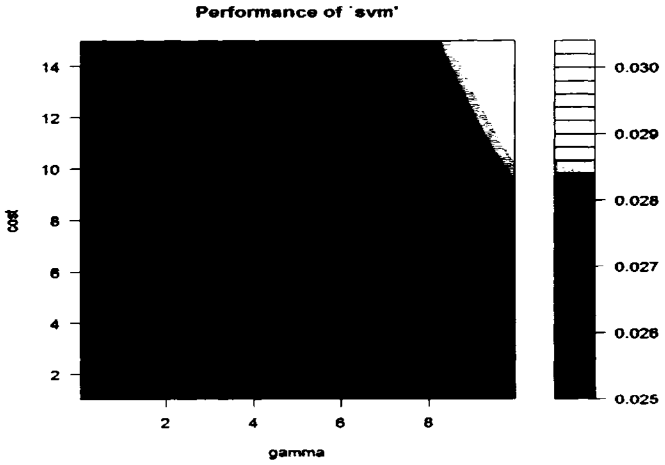


Figure 5.22: Effect of cost and kernel width variation on error (Coho)

Table 5.9 Performance of the SVM model for the various fish blocks

Data Set	Bias	Mean Absolute Error	Mean Squared Error	R^2 (in %)
Coho	-36.9	210.2	149536.9	13.9
Halibut	7.44	194.7	108833.5	12.6
Herring	-0.0003	271.11	251998.6	12.4
Humpy	-0.89	394.1	369430.2	23.9
King	-0.01	346.9	359991.5	26.5
Pink	-12.2	120.5	25780.8	5.3
Red	-8.1	341.2	317117.3	23.4
Silver	-22.1	199.1	137845.4	8.1
Tomcod	-41.1	206.3	147488.4	16.9

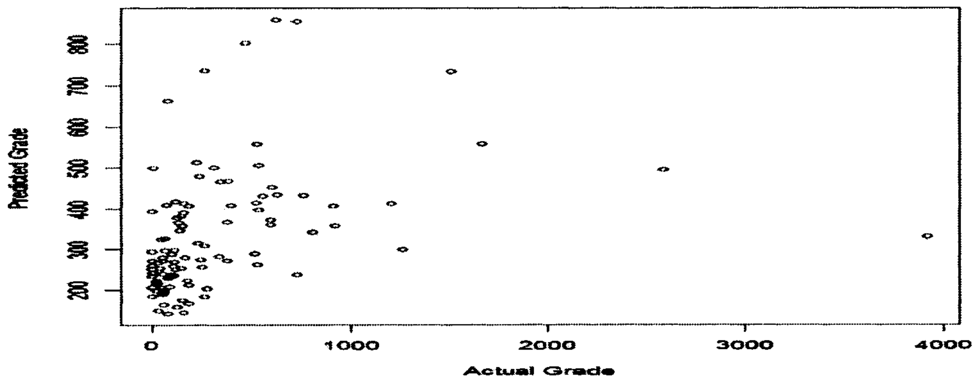


Figure 5.23: Scatter plot for actual vs. predicted grade (Herring)

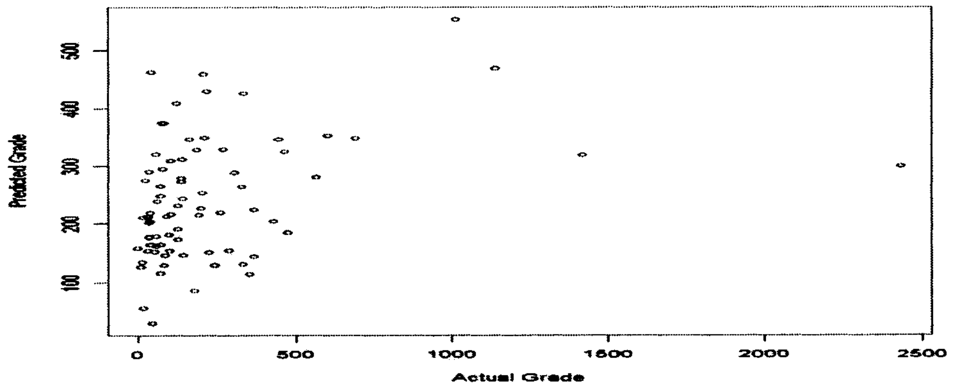


Figure 5.24: Scatter plot for actual vs. predicted grade (Halibut)

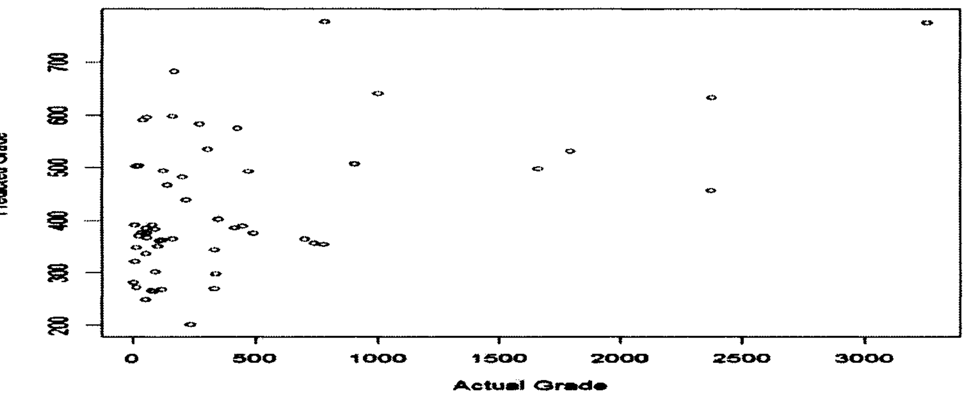


Figure 5.25: Scatter plot for actual vs. predicted grade (Humpy)

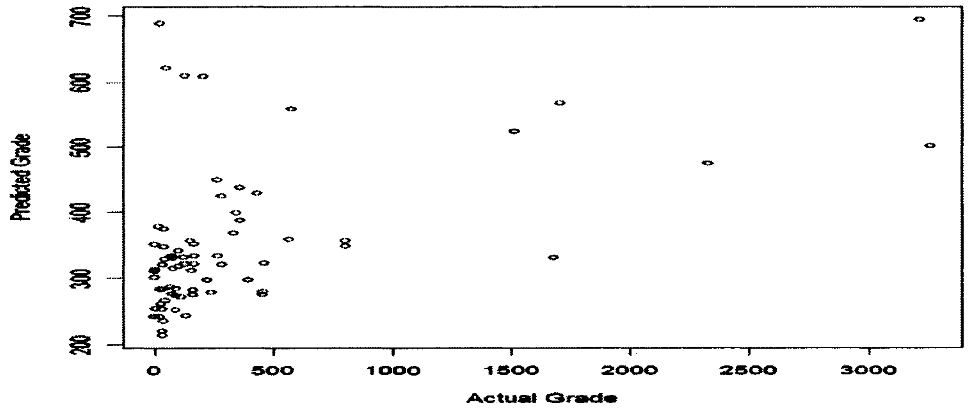


Figure 5.26: Scatter plot for actual vs. predicted grade (King)

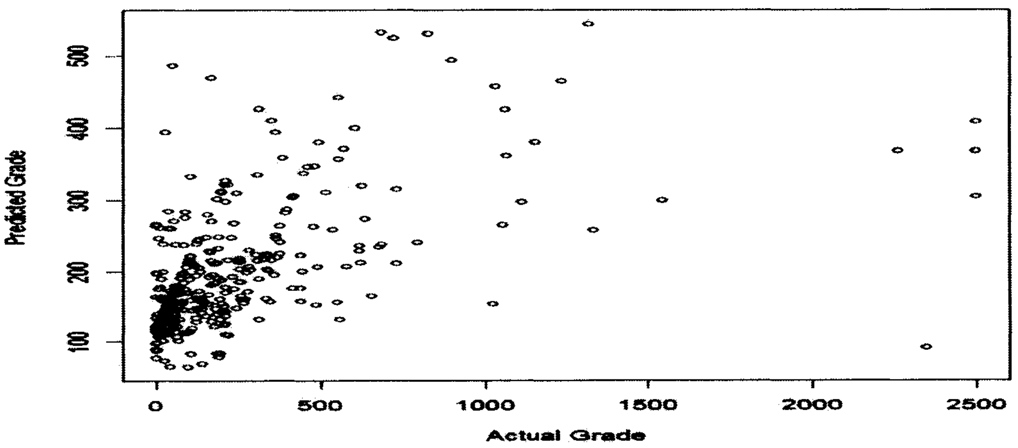


Figure 5.27: Scatter plot for actual vs. predicted grade (Pink)

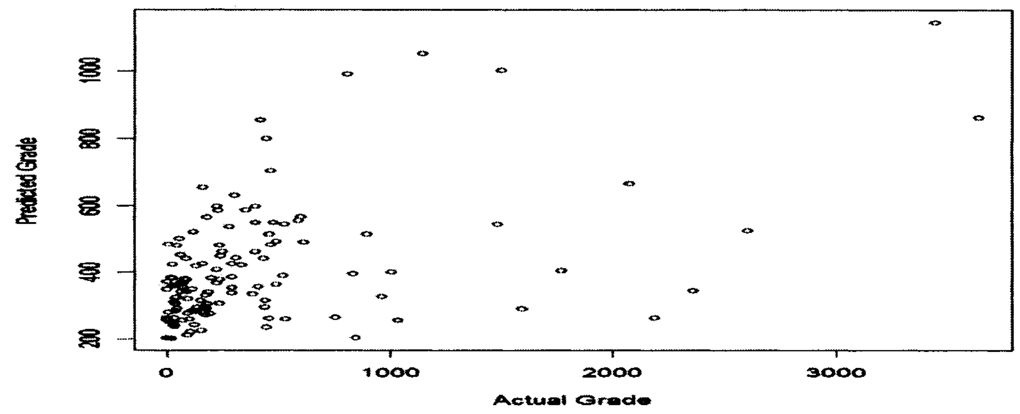


Figure 5.28: Scatter plot for actual vs. predicted grade (Red)

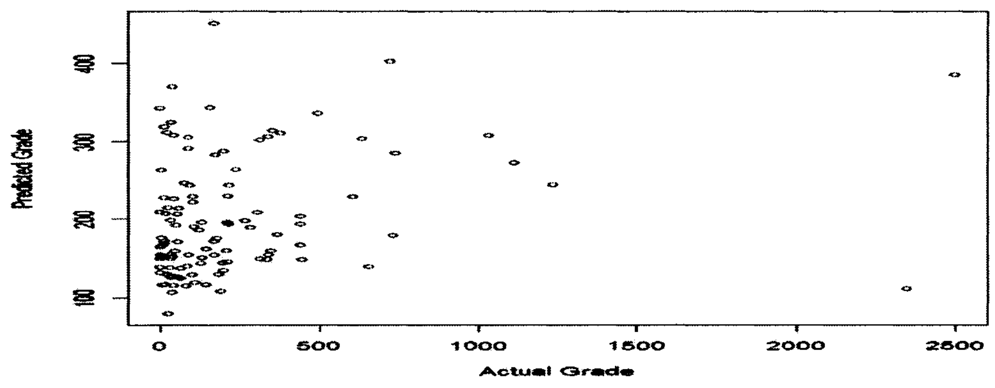


Figure 5.29: Scatter plot for actual vs. predicted grade (Silver)

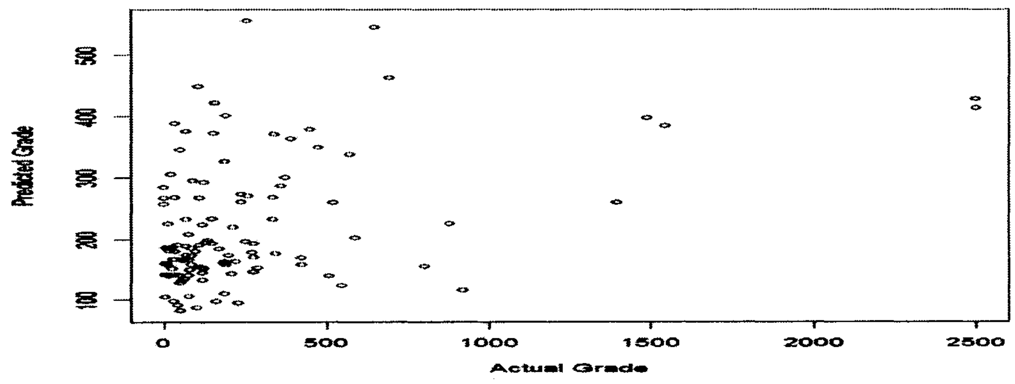


Figure 5.30: Scatter plot for actual vs. predicted grade (Tomcod)

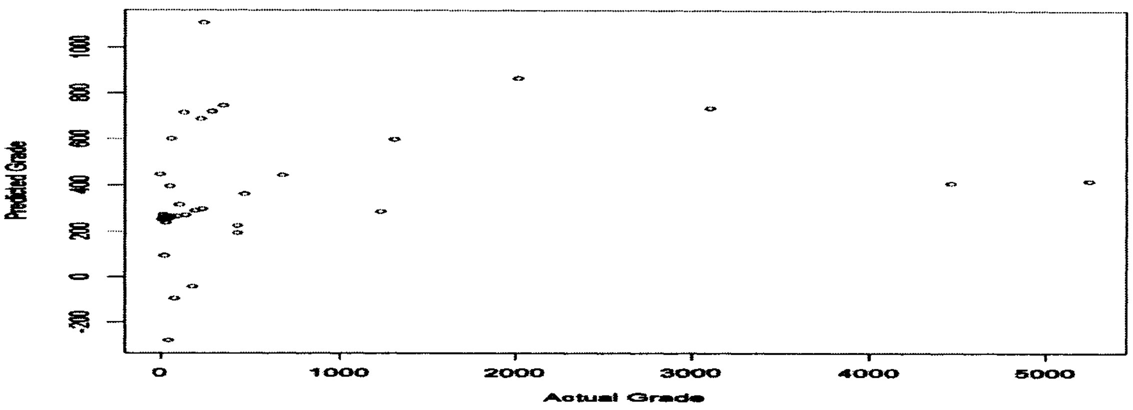


Figure 5.31: Scatter plot for actual vs. predicted grade (Coho)

5.4 Grade Forecasting and Reserve Estimation

A comparison of the two machine learning algorithms and the ordinary kriging method on the prediction dataset of the individual blocks indicates that the SVM methodology produced better estimates. It was ascertained from the high R^2 and low error values obtained with SVM as compared to the other two methods. Therefore, for estimating the ore reserve within the boundary limits of each fish block the SVM methodology was used. The reserve estimation generally involves the following three steps. The first step involves the development of a network model from the available drill hole data. The second step involves the estimation of grades at the unknown locations. These locations were carefully chosen so as to prevent extrapolation beyond the sampling domain. The final step involves the determination of the tonnage associated with the grade at each location. As mentioned in the earlier section, a SVM model with model parameters listed in Table 5.8 was used to estimate the total mineral inventory associated within the nine fish bocks. The selection of the unknown locations involved the transformation of the scattered drill hole data from each block into a grid base system. Each grid had a dimension of 25m x 25m along the northing and easting. The surface mapping system software Surfer (2002) was used for this purpose. To prevent extrapolation, the data grids outside the leased boundaries were removed. This is a two-step procedure. The first step involved the formation of grids of 25m x 25m dimension along the entire sampling region and then the removal of the grids outside of the polygonal area which enclosed the sampled points. Figure 5.32 shows one such polygonal area enclosed by the Humpy block. The grade estimation in each of the fish blocks was done for the grids that are enclosed within these polygons. Table 5.10 shows the total number of grids within the sampling region and the number of grids that are within the polygonal area. The inventory was calculated at cutoff grades of 300,400,500,600 and 700 mg/cu.m within 5m of the sediment surface.

To estimate the grade at the unknown locations, three variables, the X coordinate, the Y coordinate and the water table depth of each of the grid were used as input parameters for the model.

Table 5.10: Number of grids inside the sampling region and in polygonal area

Block	Total grids	Grids in the Polygonal region
Coho	5636	2394
Halibut	13778	7032
Herring	4260	2419
Humpy	11247	3344
King	5390	2818
Pink	6080	2661
Red	9828	3560
Silver	12390	6920
Tomcod	24252	6392

The amount of ore in each grid cell is calculated by the following equation:

Amount of Ore = 25(m) x 25(m) x 5(m) x grade (mg/cu.m) (5.10)

Once the estimations at the grids locations were done, the grade values less than the cut-off grade were neglected in the total reserve estimation. Table 5.11 shows the reserve estimate for each block. The grade values for the Pink fish block were less than 300mg/cu.m. Therefore, in the estimate column no reserve was reported for the pink block (Table 5.11). For visualization purpose of the estimated gold distributions within the sampling regions of the various fish blocks, contour plots have been presented in Figures 5.33 - 5.41. The Plots show the variation of the grades in a given fish block. From Table 5.11 it can be observed that the Coho, the King and the Red fish blocks contain adequate amount of reserve for mining operation. The reliability in the estimations is, however, low owing to the poor performance of the SVM models.

Table 5.11: Total reserve estimate for each block (in tones):

Block	COG 300	COG 400	COG 500	COG 600	COG 700
Coho	3.9188	3.8376	3.6939	3.5527	3.3556
Herring	1.6731	1.1954	0.6751	0.2648	0.015451
Humpy	4.9387	4.9387	0.3192	-	-
King	1.3805	1.2277	1.1081	0.9738	0.8127
Pink	-	-	-	-	-
Red	4.026	2.896	2.1173	1.7179	1.5234
Silver	0.7945	0.005	-	-	-
Tomcod	2.786	1.056	0.0825	0.0118	0.0022
Halibut	0.73561	0.38202	0.22787	-	-

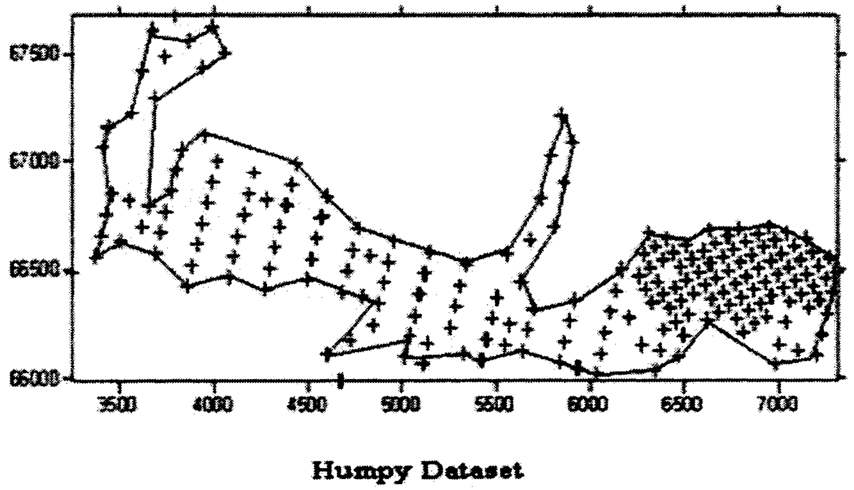


Figure 5.32: Polygonal region formed within the Humpy block.

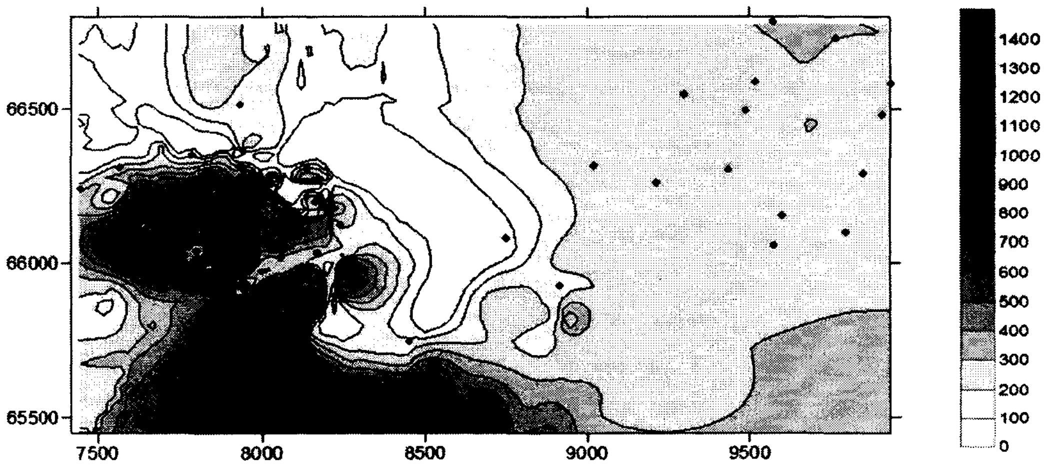


Figure 5.33: Contour plot for the predicted gold reserve (Coho)

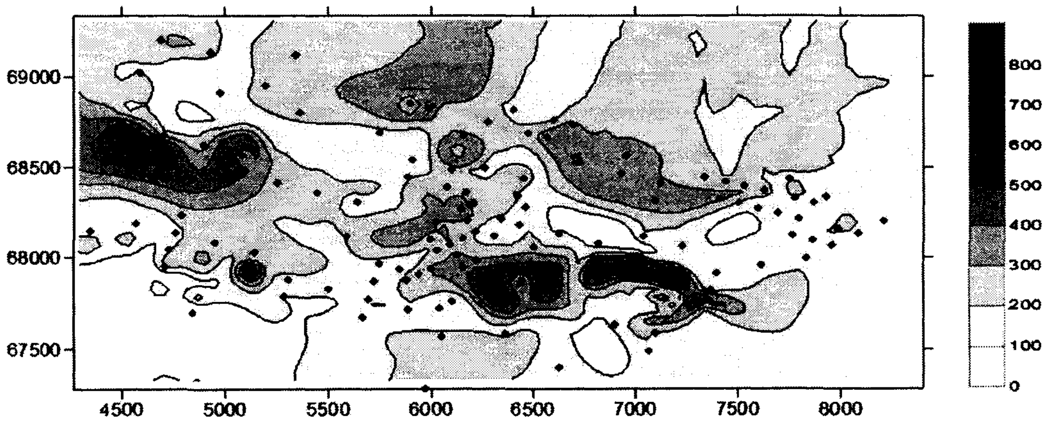


Figure 5.34: Contour plot for the predicted gold reserve (Halibut)

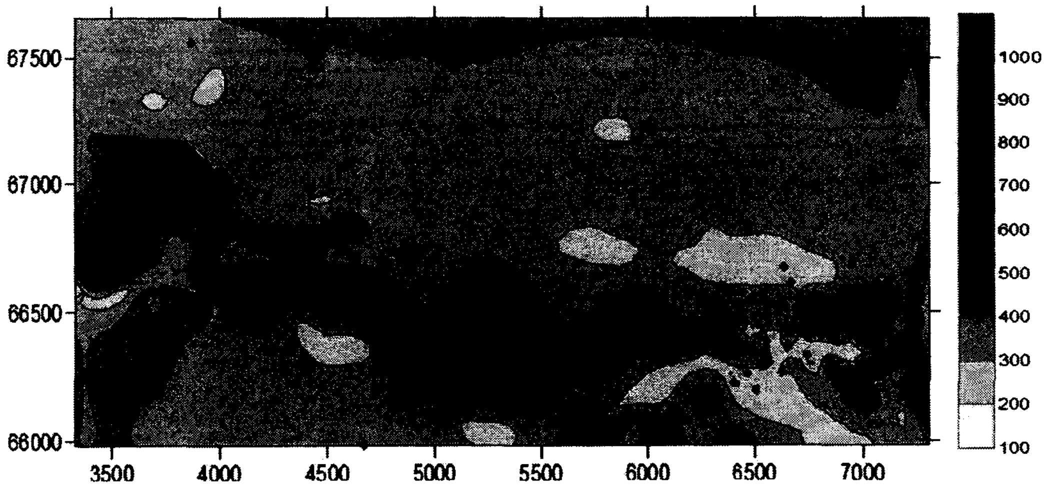


Figure 5.35: Contour plot for the predicted gold reserve (Humpy)

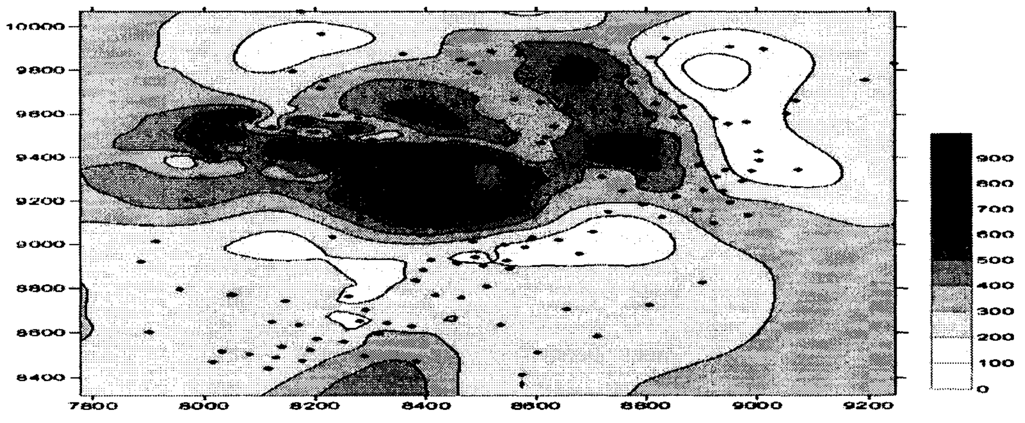


Figure 5.36: Contour plot for the predicted gold reserve (Herring)

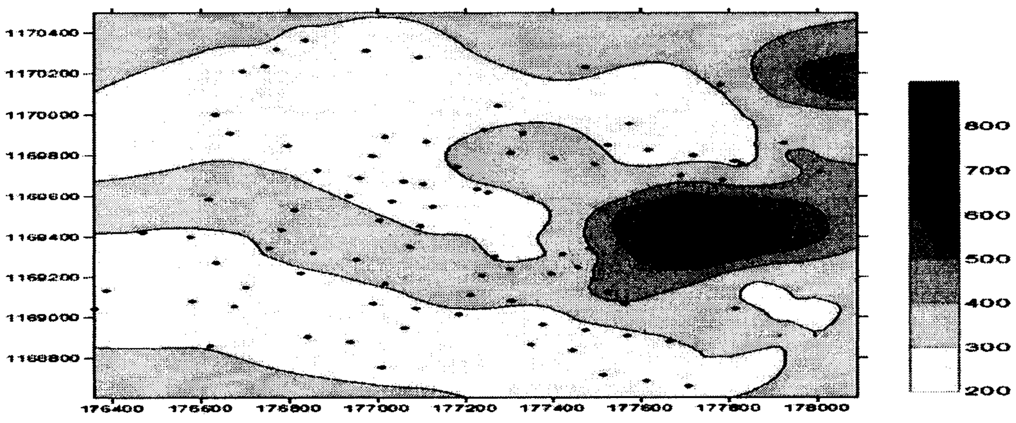


Figure 5.37: Contour plot for the predicted gold reserve (King)

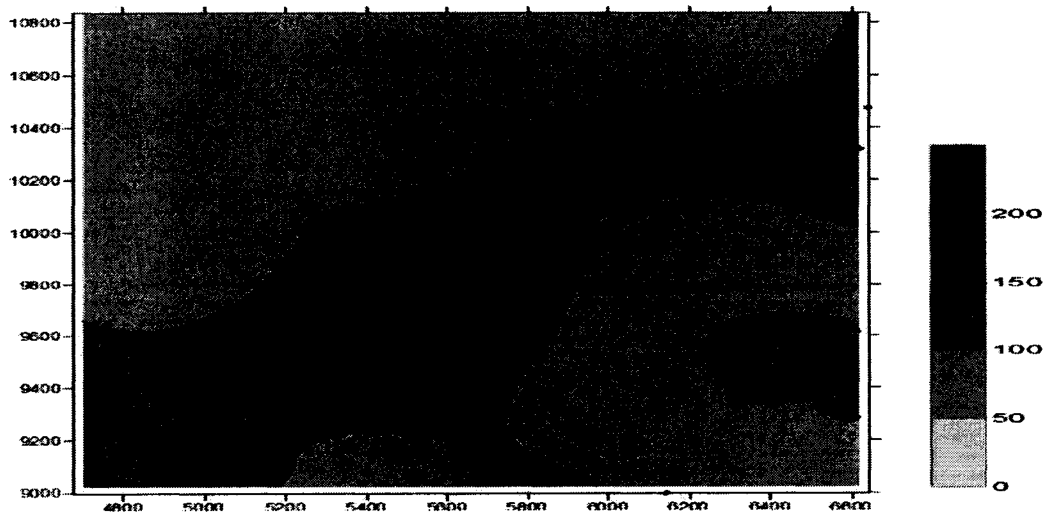


Figure 5.38: Contour plot for the predicted gold reserve (Pink)

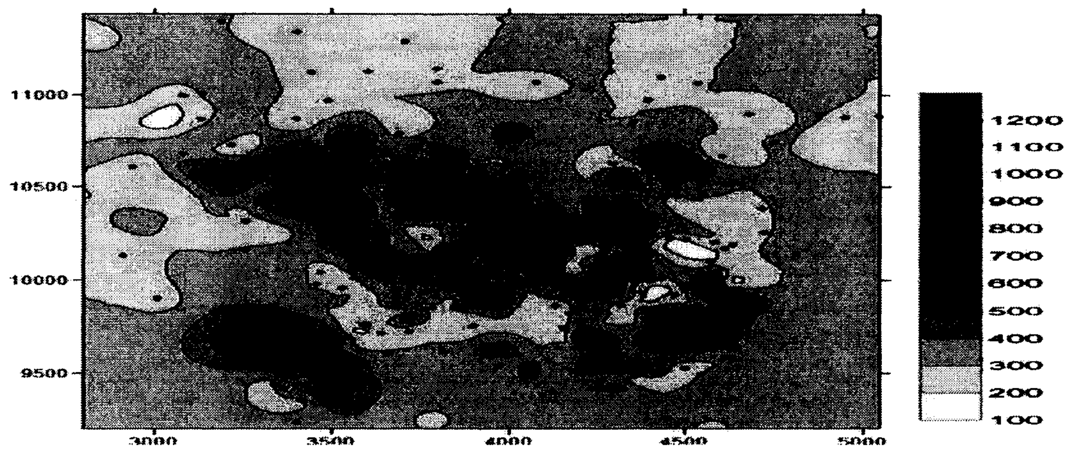


Figure 5.39: Contour plot for the predicted gold reserve (Red)

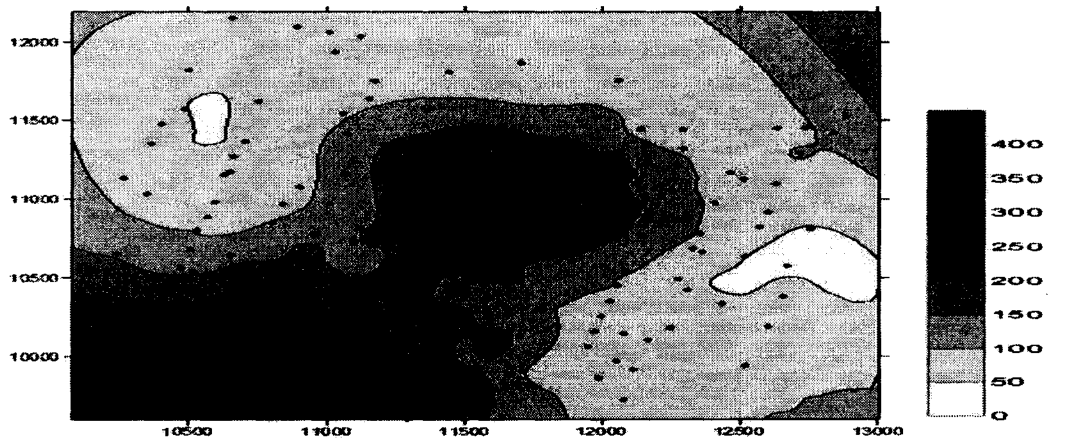


Figure 5.40: Contour plot for the predicted gold reserve (Silver)

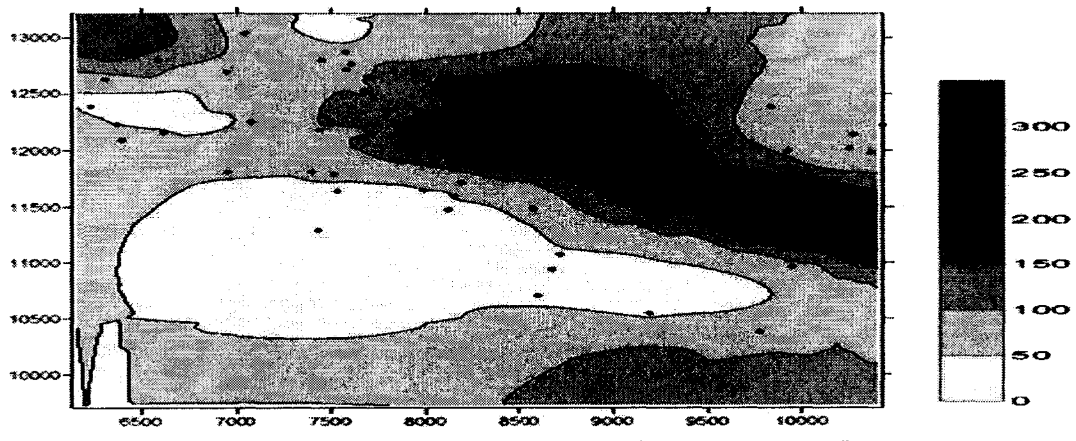


Figure 5.41: Contour plot for the predicted gold reserve (Tomcod)

CHAPTER VI
COMPARATIVE ANALYSIS OF THE ESTIMATION TECHNIQUES IN A LODE DEPOSIT

The application of the machine learning algorithms, namely the neural network (NN) and the support vector machine (SVM) on the Nome datasets (Chapter 5) resulted in an improved generalization performance for the model datasets. The performance was not only better than the traditional geostatistical ordinary kriging (OK) method but the use of these techniques also provided a comprehensive way of treating the complex phenomenon of ore reserve estimation for sparse and noisy data. The improvement, though marginal, was largely affected by the extreme spatial variations associated with the typical placer gold characteristics. In other words, the datasets dictated more than the models used for the prediction. As a result, they didn't generalize as well as one might expect and the results were difficult to analyze. This is a common problem that arises when model comparison studies are based on a single dataset. Therefore, in order to further establish these models as a reliable tool for ore reserve estimation, they have been applied to an entirely different type of deposit. The depositional environment of the deposit (lode) is devoid of the characteristics associated with the Nome placer gold deposit. An analysis similar to the Nome data was performed, i.e. the ordinary kriging, neural network and the support vector regression (SVR) were used to estimate the grade of the silver associated with the lode deposit.

6.1 Description of the Study Area

The data for the following study was obtained from the Greenscreek mine located in southeast Alaska on the northern end of Admiralty Island, approximately 25 km south of Juneau (Figure 6.1). The underground mine which is a joint venture between Kennecott Minerals and Hecla is operated by the Kennecott Greens Creek Mining Company. The orebody, which is polymetallic (silver, zinc, gold, and lead), stratiform, and a massive sulfide deposit, was discovered in 1975. The exploration drilling in the deposit began in 1978. It is hosted in Karnian/Norian-age black argillite occurring below the flood basalts and is linked to an active rift segment plumbed by sepentinite/gabbro and filled with euxenic argillite. The host rock is predominantly marine sedimentary, and mafic to ultramafic volcanic and plutonic rocks, which have been subjected to multiple periods of deformation. These deformations have imposed intense tectonic fabrics on the rocks as a result of which the mineralization occurs discontinuously along the contact between a structural hanging wall of quartz mica carbonate phyllites and a structural footwall of graphitic and calcareous argillite. Major sulfide minerals are pyrite, sphalerite, galena, and tetrahedrite/tennanite (www.hecla-mining.com/propGreens.html).

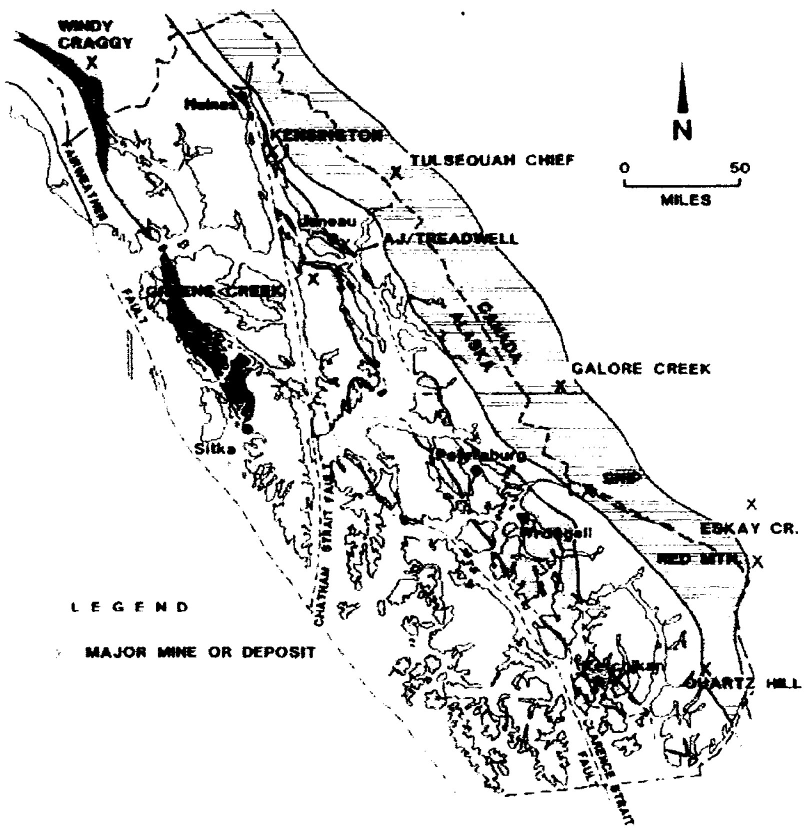


Figure 6.1: Location of the Greens Creek Mine, Alaska (from http://www.kennecottminerals.com/S&E_2002/GreensCreek.pdf)

6.2 Model Development and Analysis of the Data

The data used for the study consisted of 432 exploratory boreholes. The data is available in terms of easting (x) and northing (y) co-ordinates (in m), gold, silver, lead, zinc and copper content (in ppm). A statistical analysis of the data indicated that the mean and the standard deviation values are 0.96 ppm and 1.32 ppm of silver. Visual examination of the histogram plot indicates that silver follows an approximately lognormal distribution (Figure 6.2). Spatial continuity of the silver values was also explored through the variography study. Figures 6.3 presents a snapshot of the semi-variogram modeling using the log-transformed data of silver values. Since a lognormal distribution of the silver values is assumed, it was decided to incorporate lognormal kriging to treat this variable. This choice was further supported by an independent comparative study of lognormal kriging with ordinary kriging which indicated that lognormal kriging produced better estimates than the ordinary kriging.

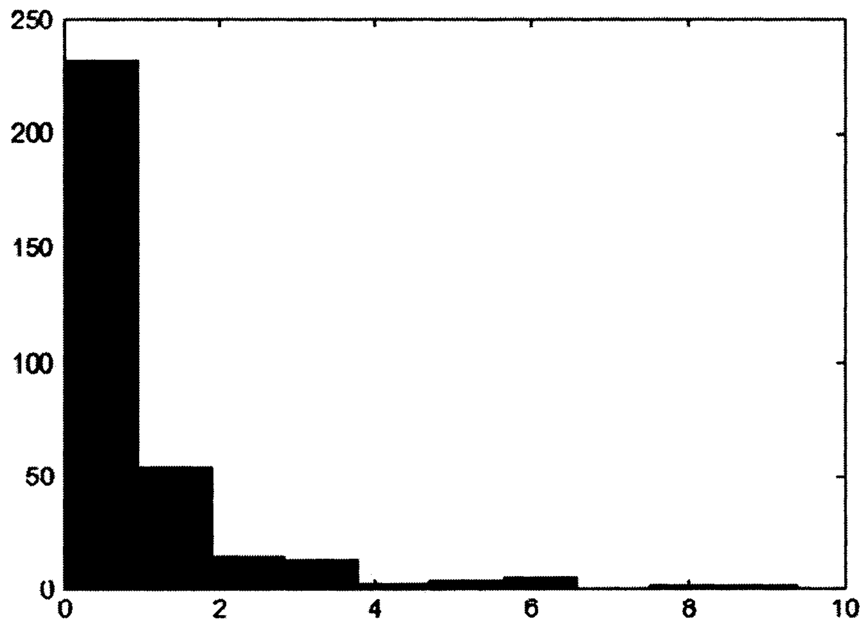
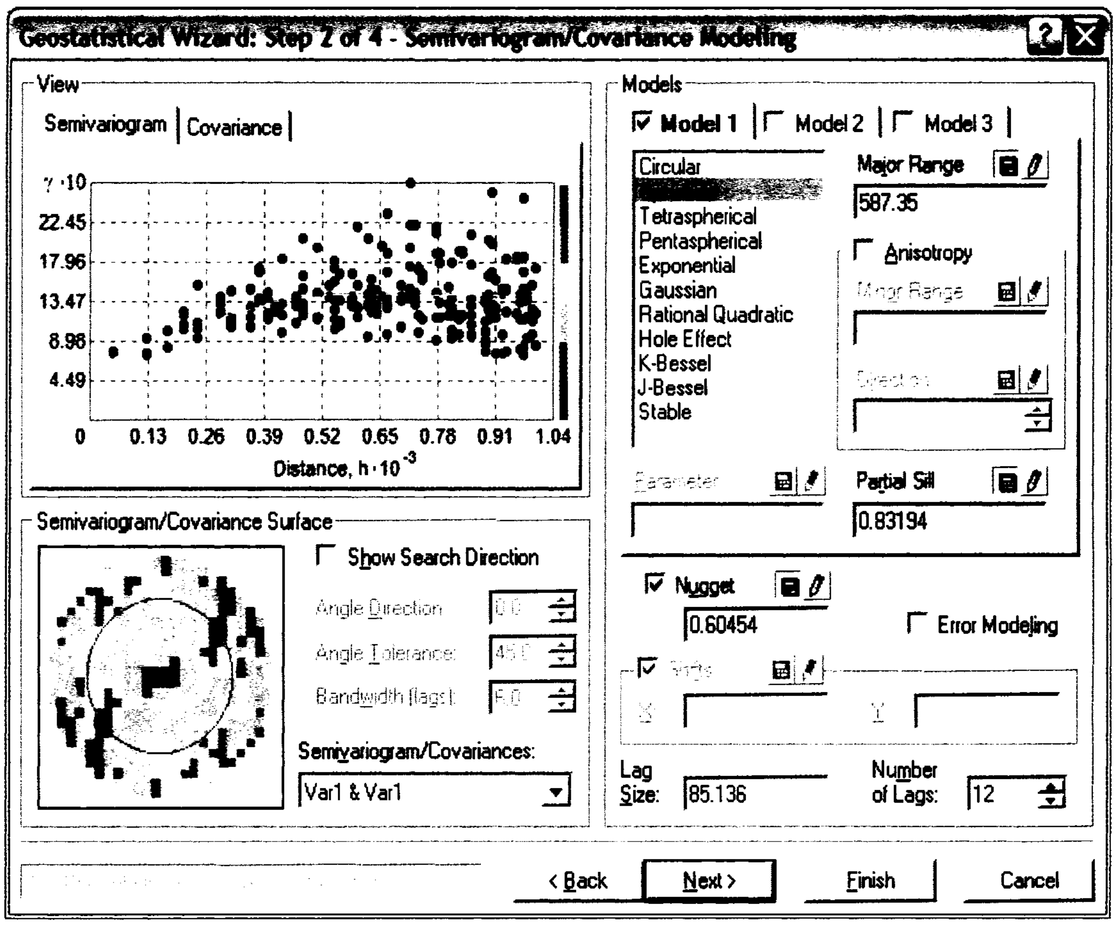


Figure 6.2: Histogram plot for the Silver values



Geostatistical Wizard: Step 2 of 4 - Semivariogram/Covariance Modeling

View: Semivariogram | Covariance

Semivariogram/Covariance Surface

☐ Show Search Direction
Angle Direction: 0.0
Angle Tolerance: 45.0
Bandwidth (lags): 6.0

Semivariogram/Covariances: Var1 & Var1

Models: ☑ Model 1 | ☐ Model 2 | ☐ Model 3

Circular
Tetraspherical
Pentaspherical
Exponential
Gaussian
Rational Quadratic
Hole Effect
K-Bessel
J-Bessel
Stable

Major Range: 587.35
☐ Anisotropy
Minor Range:
Direction:
Parameter:
Partial Sill: 0.83194

☑ Nugget: 0.60454
☐ Error Modeling
☑ Shift: X: Y:

Lag Size: 85.136
Number of Lags: 12

< Back | Next > | Finish | Cancel

Figure 6.3: Snapshot of the semi-variogram modeling on the variable Silver

The major difficulty with any ore reserve estimation method is the inability to verify if the predicted value bears a close resemblance with the actual one. Before actual production in a mine, the only reliable way to verify the reliability of the estimation method is to test the actual value with the predicted value using a data set which was not used previously for training of the model. Therefore, prior to the model development, the available data was divided into three statistically similar subsets employing the genetic algorithms (GA). The training dataset constituted half of the samples while the calibration and the prediction dataset constituted the remaining half. Table 6.1 shows the statistical summaries of the three datasets in terms of the mean and the standard deviation values. It could be seen that the values are in close agreement for the three datasets.

Table 6.1: Statistical Properties of the Greens Creek model datasets.

Training Dataset	Mean	SD
X	5541.924316	409.3363647
Y	3752.673828	541.2883911
Gold	0.025	0.0572633
Lead	0.151389	0.2821809
Zinc	3.4138887	7.3849754
Copper	2.8874996	3.9141705
Silver	0.9583336	1.4291731
Calibration Dataset		
X	5558.49707	416.8015442
Y	3707.911133	494.9816589
Gold	0.0259259	0.0536001
Lead	0.1342593	0.2271982
Zinc	3.740741	5.0234962
Copper	2.7462976	3.3884475
Silver	0.9185182	1.179113
Validation Dataset		
X	5567.846191	429.4827576
Y	3670.762695	520.6760254
Gold	0.0277778	0.056093
Lead	0.1398148	0.2557525
Zinc	3.0342593	4.0313764
Copper	2.689815	3.529341
Silver	0.8907409	1.202143

For the NN model development the training dataset was used to train the model while the prediction dataset was used to assess the model generalization ability. During this process the calibration dataset served as an independent observer of the entire training process as described in chapter 5. Since there was no need of a calibration dataset in SVR and OK modeling approach, the calibration dataset was merged with the training dataset in order to select the optimal model parameters.

For NN modeling, the commercially available software package Neuroshell, 1993 was used. Various network architectures with different numbers of hidden layers and neurons in each layer were investigated prior to the selection of a slab architecture as shown in Figure 6.4. The purpose behind this modeling exercise was to ensure that the model is neither overfitted nor underfitted. Overfitting of a NN model is a condition which arises when there are too many neurons in the hidden layer as a result the network performs exceptionally well in the training dataset but doesn't generalize well. On the other hand, underfitting is a condition arising due to less number of neurons during which the network results in high training and high generalization error. For this modeling exercise, the network comprised of 5 slabs: one input slab, 3 hidden slabs and 1 output slab (a slab is basically a group of neurons; a particular layer may have multiple slabs). Each slab in the hidden layer and the output layer consisted of different activation functions. The input slab has six neurons for each of the input variables while the output slab has one neuron for the silver values as the output variable. The slabs in the hidden layer have 8, 6 and 8 neurons respectively. The three slabs in the hidden layer use three different activation functions viz. tanh, gaussian and complementary gaussian whereas the output layer slab uses a linear activation function. The concept behind using different combinations of the activation functions is to identify various patterns in the dataset. A particular activation function may be more suitable for a few typical patterns; however, it may not work at all for others patterns. Thus, the use of different activation functions ensures that at least some of the underlying trends in the data are captured. For example, a gaussian activation function in one hidden slab may detect features in the mid-range of the data while a gaussian complement activation function in another hidden slab may detect features from the upper and the lower extremes of the data. Similarly, a tanh activation function will tend to group together data at the low and the high ends of the original data range. This may be helpful in reducing the effects of outliers. Implementation of these features in the output layer may result in better predictions. The characteristic equation of the output signal from the gaussian, complementary gaussian and tanh activation function are given in equations 6.1-6.4.

$$\text{Gaussian activation function} = \exp(-x^2) \tag{6.1}$$

$$\text{Gaussian-complement activation function} = 1 - \exp(-x^2) \tag{6.2}$$

$$\text{Tanh activation function} = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} + \exp^{-x}} \tag{6.3}$$

$$\text{Linear activation function} = x \tag{6.4}$$

Where, x is the input to the activation function.

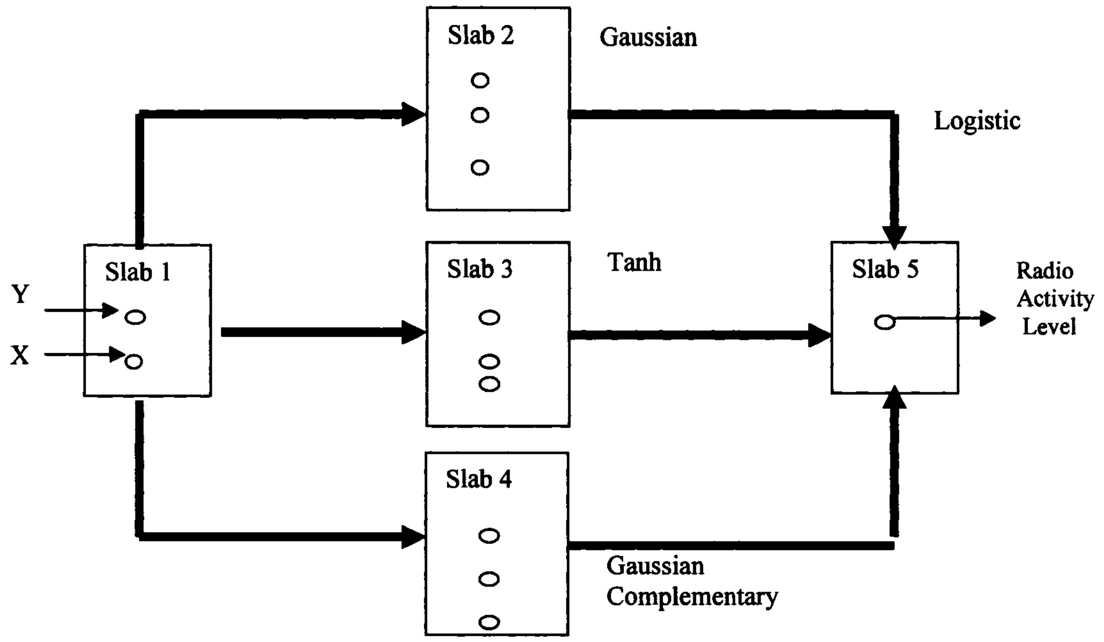


Figure 6.4: Ward net architecture for the NN modeling

For the SVM modeling, a grid based approach with 10 fold cross validation on the training dataset was employed to select the optimal model parameters C, σ and ε. Figure 6.5 shows the plot for the model performance (troughs and flat regions) for different combinations of the C and σ values. The cross-validation MSE was used as a criterion to select the optimum parameter values of C, σ and ε. The flat regions correspond to the various possible combinations for the optimal values of C and σ. The optimal estimates of C and σ were found to be 2.5 and 0.5 respectively.

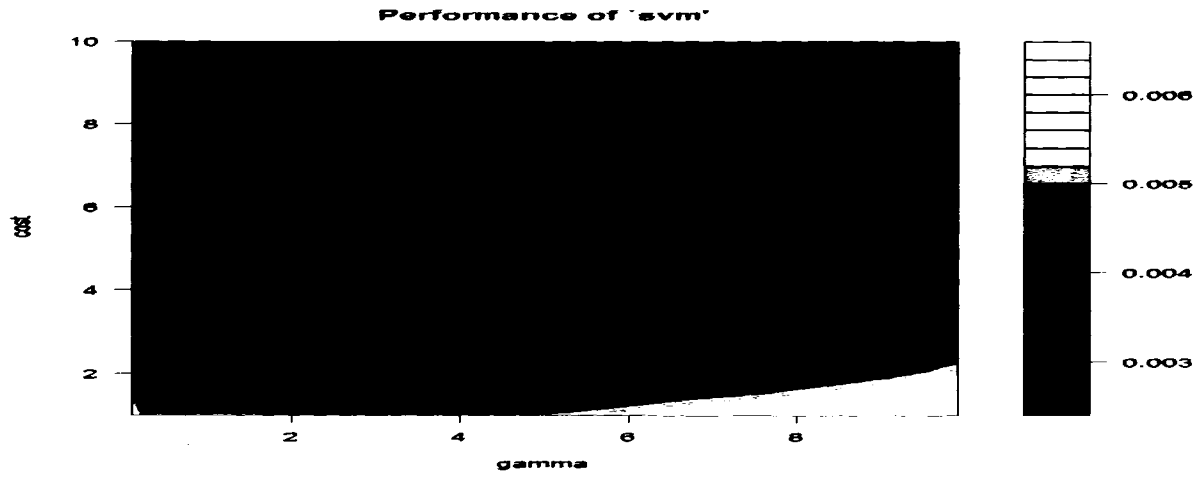


Figure 6.5: Effect of the cost and kernel width on the error for the Silver values

Once the optimum values of these parameters were determined the next step involved the selection of an optimum value of ε. This was selected by fixing the values of C and σ at their optimum values, while varying the parameter ε. This exercise was also carried out through a cross-validation study on the training data set. Figure 6.6 shows the variation of the mean squared error with respect to the parameter ε for the training dataset.

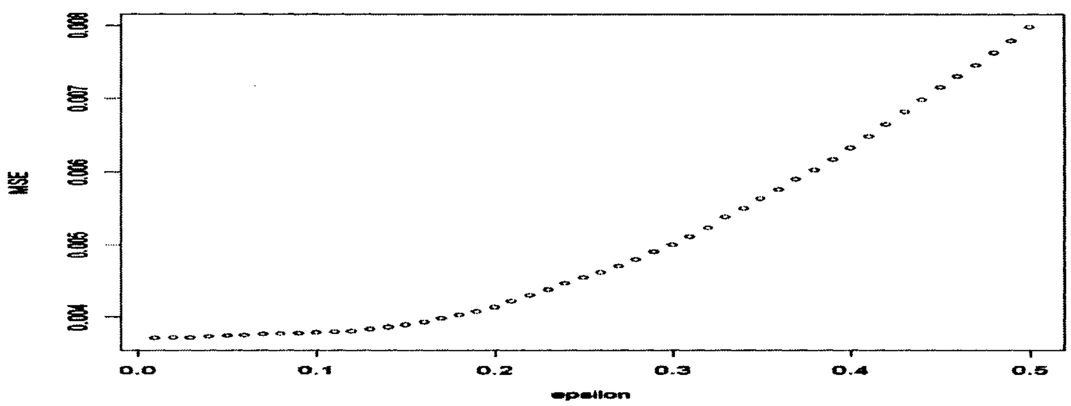


Figure 6.6: Variation of error with epsilon ε for the variable Silver.

Following this exercise, the optimum model parameter values of C, σ and ε for the silver values were found to be 2.5, 0.5 and 0.05 respectively. The final step involved the assessment of the model generalization ability through the examination of the generalization error on the prediction data set.

6.3 Results and Discussion

An independent prediction data set consisting of 108 observations was used to validate the overall performance and the generalization capability of the OK, NN and SVM models. Table 6.2 presents the performance of the individual models on the prediction dataset. The various criteria that were used to compare the model performance was the mean error (ME), the mean absolute error (MAE), the root mean squared error (RMSE) and the coefficient of determination (R^2). Based on these, a summary statistic was developed as a measure of performance. This summary statistic, termed the "skill value," is an entirely subjective measurement and is expressed by equation 6.5. Numerous skill measures can be devised; however, the one proposed considers the ME, MAE, RMSE equally and applies a scaling to the R^2 so that it is of the same order of magnitude as the others. It should be noted that the lower the skill value, the better the method is. In this way various methods can be ranked based on their skill values i.e. by their overall performance on the prediction dataset.

$$\text{'skill value'} = \text{abs (ME)} + \text{MAE} + \text{RMSE} + 100 * (1-R^2) \tag{6.5}$$

Table 6.2: Generalization performance of the models for the variable Silver

Statistics (Silver)	SVM	NN	OK
Mean Error	0.02	0.08	0.25
Mean Absolute Error	0.25	0.36	0.64
Root Mean Squared Error	0.48	0.72	1.04
R^2	0.91	0.79	0.59

Table 6.3 presents the skill values and the ranks for the various methods that were used on the prediction dataset. It could be seen from the table that the machine learning algorithms performed significantly better than the traditional kriging method. The difference in the skill values is mainly due to the high variation in the R^2 (Table 6.2).

Table 6.3: Model performances based on skill values

Statistics (Silver)	SVM	NN	OK
skill value	9.75	22.16	42.93
Rank	01	02	03

Figures 6.7–6.9 show the scatter plots of the true vs. predicted grade values for variable Silver. It could be seen from the plots that the SVM method overperforms compared to the other two methods. To further investigate the performance of the model, the errors in the prediction set were analyzed. Figures 6.10-6.12 show the prediction error distribution plots for the OK, NN and the SVM methods. It can be noted that error distribution of Silver values for the SVM model and the NN model approximates a normal distribution, whereas for the OK method it is more of a lognormal shape. A normality assumption of the model errors is always preferred. Thus, a lognormal error

distribution of the OK method could be seen as a disadvantage. This is particularly significant where uncertainty analysis is conducted.

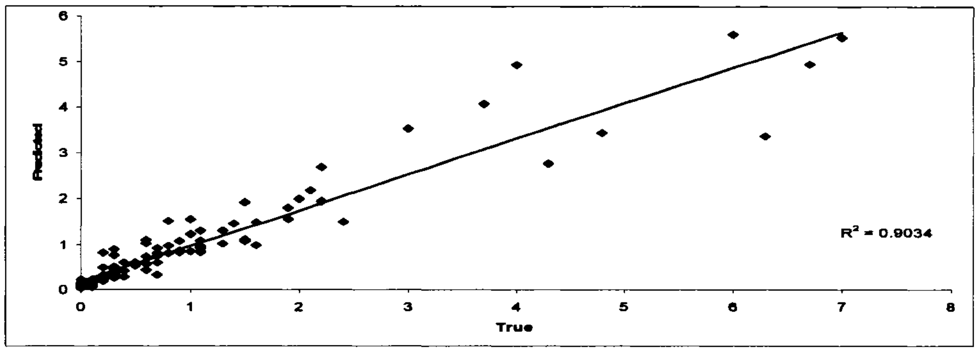


Figure 6.7: True vs. predicted (SVM)

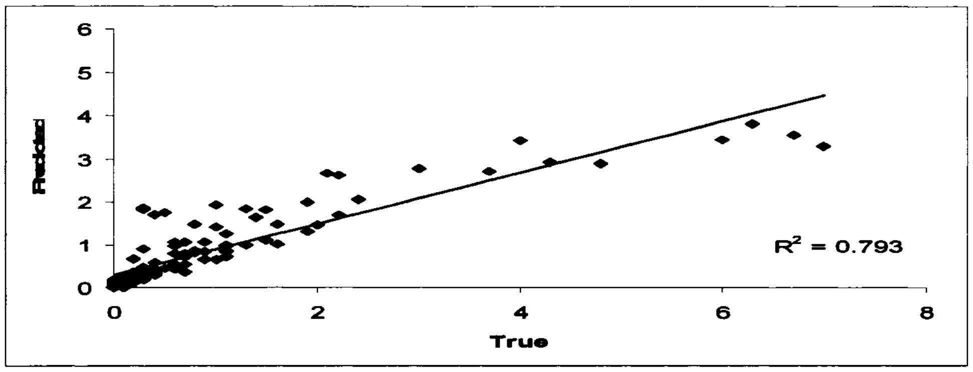


Figure 6.8: True vs. predicted (NN)

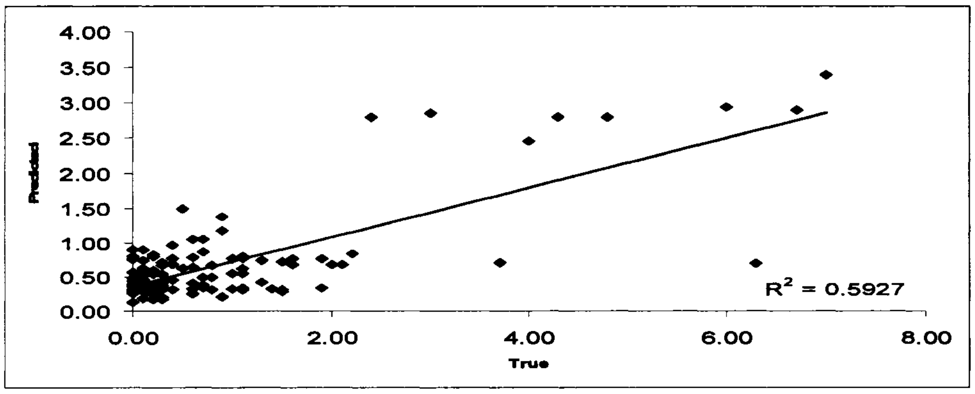


Figure 6.9: True vs. predicted (OK)

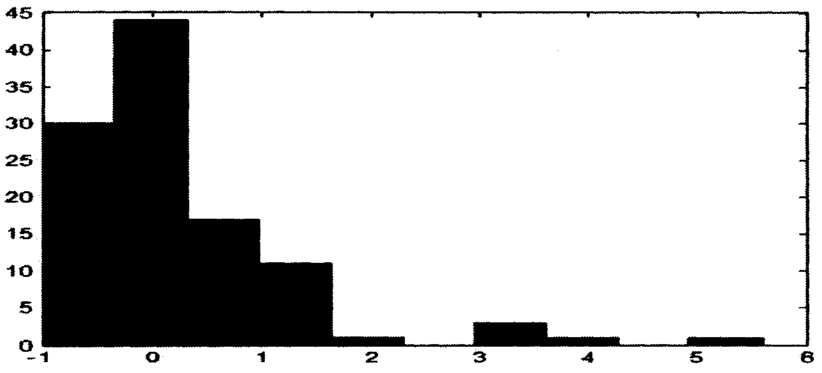


Figure 6.10: Error distribution for the Silver values (OK)

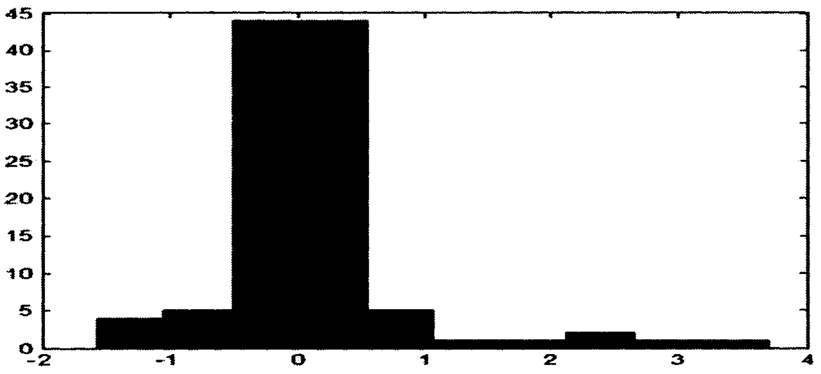


Figure 6.11: Error distribution for the Silver values (NN)

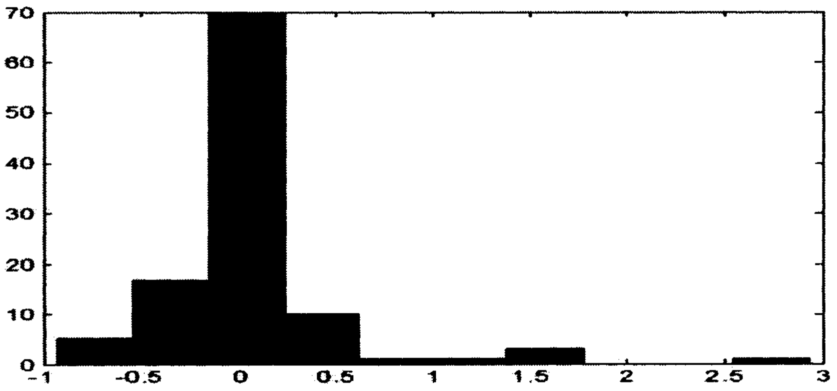


Figure 6.12: Error distribution for the Silver values (SVM)

CHAPTER VII

ESTIMATION OF GOLD RESERVE BY SUPPORT VECTOR MACHINES: DATA DIVISION USING CLUSTERING ALGORITHMS

In Chapter 5, the methodologies of OK, NN and SVM were applied on the Nome data after a manual division of the entire dataset into nine fish blocks. A major objective during data division was to divide the data such that *(a)* the data within each block are spatially correlated, *(b)* the dataset of any two blocks are not spatially correlated and, *(c)* each of the data blocks are spatially discrete and does not overlap with another. The division of the data into fish blocks (*Coho, Halibut, Herring, Humpy, King, Pink, Red, Silver* and *Tomcod)* was done arbitrarily. From a mining point of view, these arbitrary fish blocks do not make much sense. The lease boundary could be divided based on the concentration of ore grade values and could be clustered differently. In this chapter, clustering algorithms were used to divide the entire dataset into an optimal number of clusters. Each cluster can be identified as a mining block. The SVM methodology was then applied separately on each of the clusters. Two contemporary clustering algorithms were considered for the data division: the Fuzzy C-means algorithm (Chung and Lee, 1992; Hoppner et al., 1999) and the K-means algorithm (Forgy, 1965; Hartigan and Wong, 1979).

7.1 Clustering Algorithms

Analysis of the results involved two steps:

- Clustering the data set in to *k* clusters using the spatial coordinates and the gold estimates as the influencing variables (feature vectors).
- Using support vector machines on each of the clusters to estimate the gold grade.

7.1.1 K-Means Algorithm

The k-means clustering algorithm is well known in published literature. Its computational complexity is a function of the number of data points (feature vectors) to be clustered. The algorithm to perform k-means clustering in a set of data is described below:

1. Consider a set of n data points (feature vectors).
2. Assume the number of clusters, k, $2 \leq k \leq n$.
3. Randomly select k initial cluster center locations.
4. All data points are assigned to a partition, defined by the nearest cluster center.
5. The cluster centers are moved to the geometric centroid of the data points in their respective partitions.
6. Repeat from (4) until the overall objective function is smaller than a given tolerance or the centers do not move to a new point.

The k-means clustering algorithm tries to minimize an objective function. In this case, it is a squared error function. The data is clustered by the k-means method, which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centers is minimized. At the minimum, all cluster centers are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre). The objective function can be expressed as:

$$J=\sum_{j=1}^{k}\sum_{I=1}^{n}\left\|x_i^{(j)}-c_j\right\|^2 \qquad (7.1)$$

where $\left\|x_i^{(j)}-c_j\right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j and is an indicator of the distance of the n data points from their respective cluster centers.

Although it can be proved that the procedure will always converge, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The k-means algorithm can be run multiple times to lessen this affect (Hartigan and Wong, 1979).

7.1.2 Fuzzy C-means Algorithm

Fuzzy C-means Clustering (FCM), also known as Fuzzy ISODATA, is a clustering technique which is separated from hard k-means that employs hard partitioning. The FCM employs fuzzy partitioning such that a data point can belong to all groups with different membership grades between 0 and 1. FCM is an iterative algorithm. The aim of FCM is to find cluster centers (centroids) that minimize a dissimilarity function.

In fuzzy partitioning, the membership matrix (U) is randomly initialized according to the following equation:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \ldots, n \tag{7.2}$$

The dissimilarity function which is used in FCM is given as:

$$J(U, c_1, c_2, \ldots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}{}^m d_{ij}{}^2 \tag{7.3}$$

Where u_{ij} is between 0 and 1; c_i is the centroid of cluster i; d_{ij} is the Euclidian distance between i_{th} centroid (c_i) and j_{th} data point; m ϵ [1,∞] is a weighting exponent. To reach a minimum of dissimilarity function, two conditions must be satisfied. These are given in the following equations:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}{}^m x_j}{\sum_{j=1}^{n} u_{ij}{}^m} \tag{7.4}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}} \tag{7.5}$$

Various steps involved in fuzzy C-means algorithm is presented below.

1. Randomly initialize the membership matrix (U) that has constraints.
2. Calculate the centroids (c_i).
3. Compute the dissimilarity between the centroids and the data points. Stop if its improvement over previous iteration is below a threshold.
4. Compute a new membership matrix, U. Go to step 2.

By iteratively updating the cluster centers and the membership grades for each data point, FCM iteratively moves the cluster centers to the "right" location within a data set. FCM does not, however, ensure that it converges to an optimal solution. It is because the cluster centers (centroids) are formed using U that are randomly initialized. The performance depends on the initial centroids. For a robust approach an algorithm can be used to determine all of the centroids (for example: arithmetic means of all data points), or FCM can be executed several times each starting with different initial centroids. In this study, a robust clustering of the data set was defined using an algorithm to determine all of the centroids.

7.2 Results and Discussion

As previously noted the k-means and the FCM clustering algorithms were used to cluster the entire dataset into a pre-defined number of clusters. The SVM methodology was then applied on each of the clusters to assess the utility of the clustering algorithms for data division. The number of clusters was varied from 2 to 16. The optimal number of clusters was determined using the criteria as follows: (a) SVM model performance on each of the clusters, (b) the spatial integrity of the data in each of the clusters.

Several measures of error such as Root Mean Square Error (RMSE), Absolute Error (AE) and coefficient of determination (R^2) were used to determine the best clustering algorithm and the corresponding optimal number of clusters. As described in Chapter 5, a 10-fold cross validation technique was used to train and to validate the SVM model for each of the clusters. With increasing number of cluster, the clustering algorithms tend to create clusters that overlap with each other.

The entire dataset was segregated into a predefined number of clusters using the k-means and FCM clustering algorithms. Figure 7.1 and Figure 7.2 show the clustered datasets obtained using the k-means and the FCM clustering algorithm respectively. It is clear that k-means clustering algorithm tends to generate overlapping clusters when the number of clusters increases beyond ten clusters.

On the other hand, the FCM clustering tends to create spatially discrete clusters for any number of predefined clusters. The stability of FCM clustering algorithm is further evident in Figure 7.3.

The distribution of the entire dataset among different clusters for varying number of predefined clusters can also be observed from Figure 7.3. It is interesting to note that the FCM clustering method indicates a better distribution of the dataset compared to the k-means. Considering the spatial discreteness of the clusters, the FCM clustering algorithm has performed better than the k-means clustering algorithm.

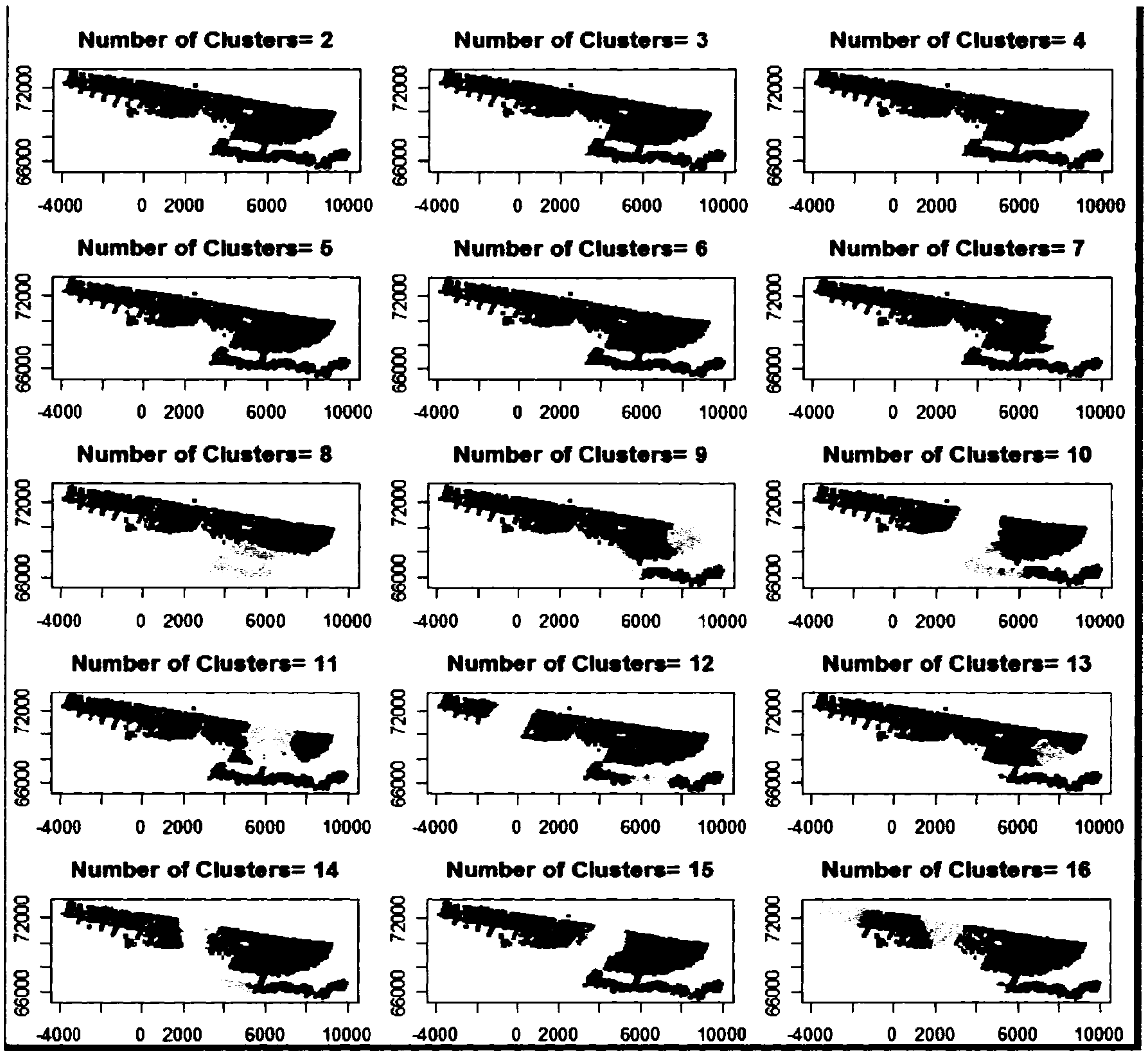


Figure 7.1: Map showing clustered dataset using K-means algorithm

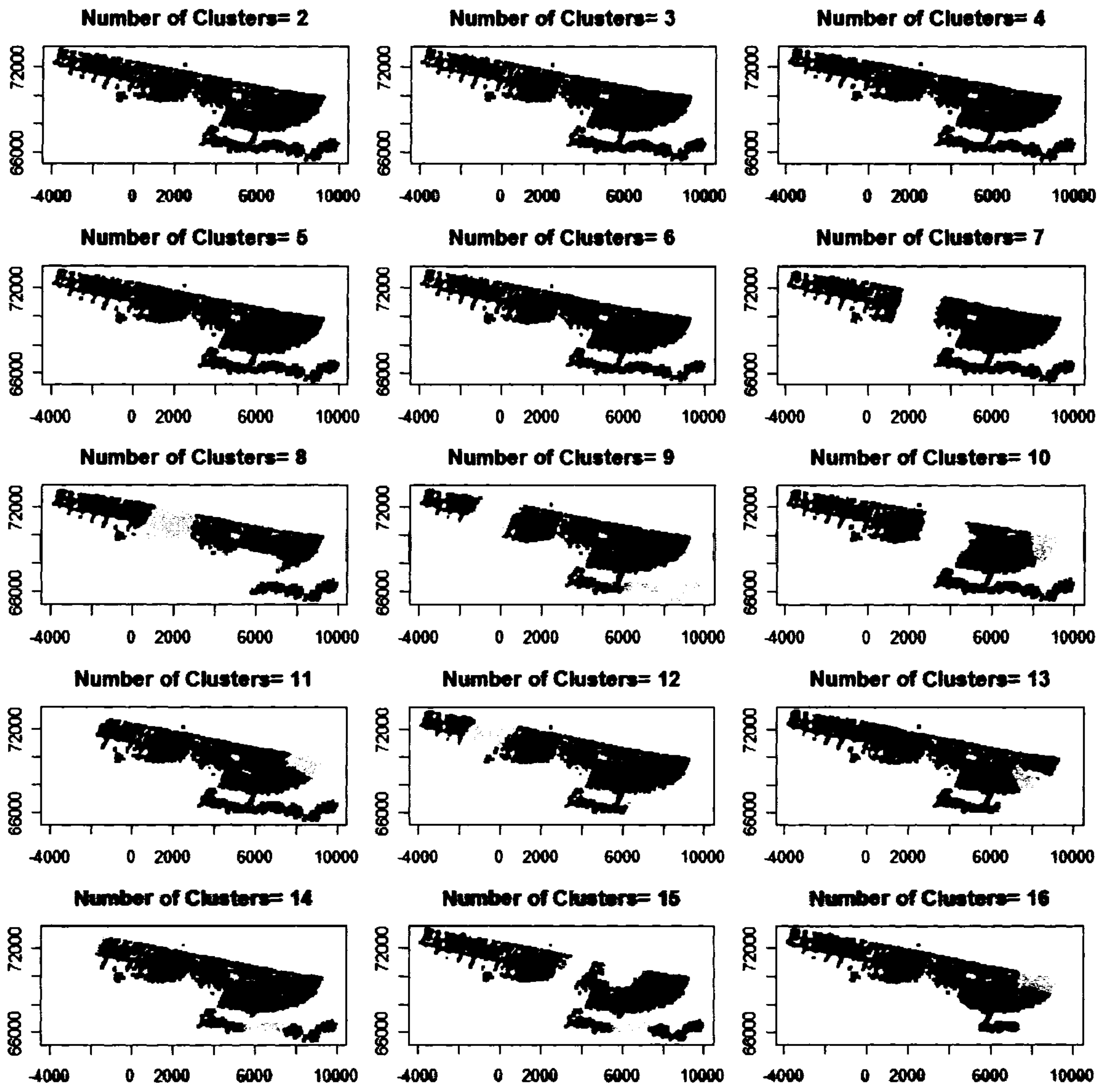
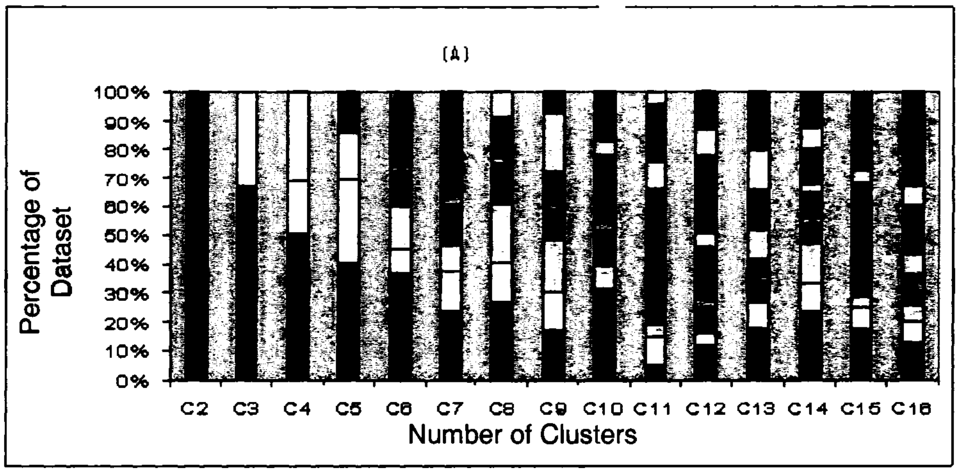


Figure 7.2: Map showing clustered dataset using FCM algorithm.

Another interesting point is the comparison between the clusters generated using the clustering algorithms and the fish blocks created arbitrarily. It may be observed that even though the trend in data division is the same, the division itself varies significantly.



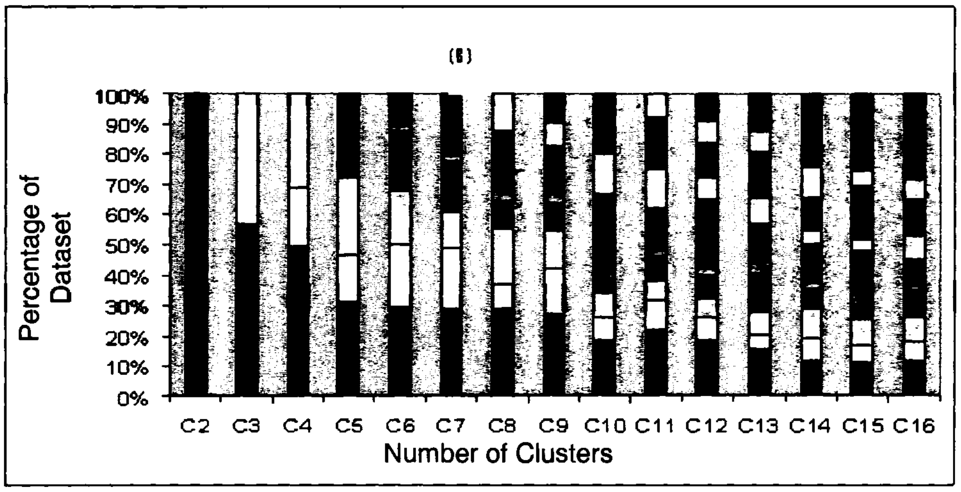
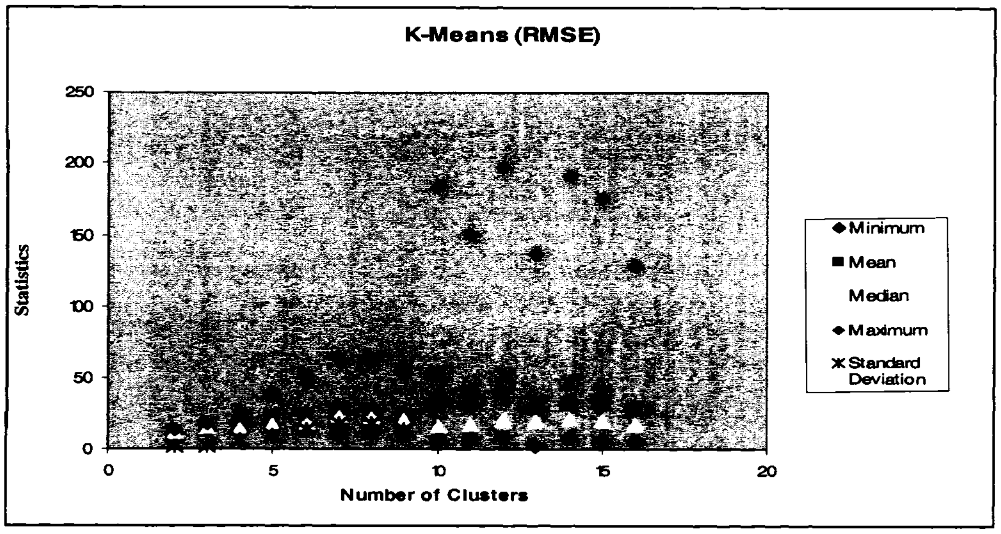
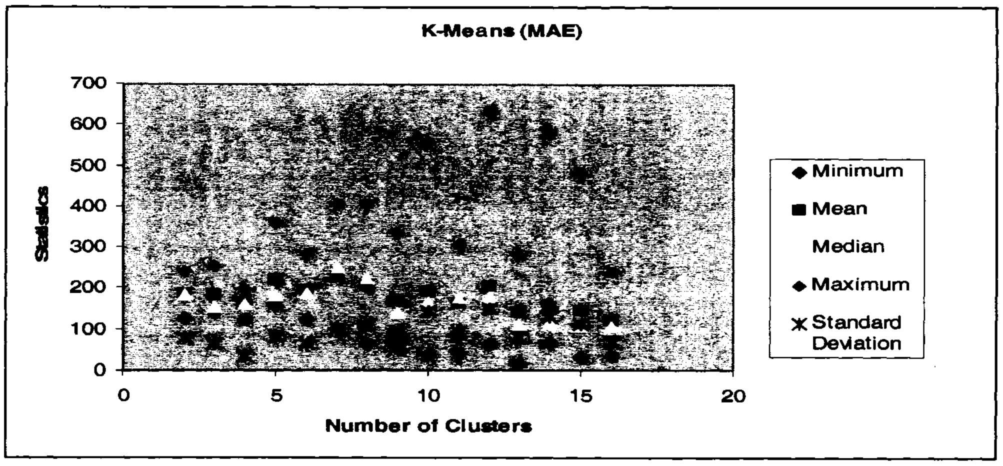


Figure 7.3. Partitioning of the dataset in clusters using (a) K-means algorithm and, (b) Fuzzy C-means algorithm.





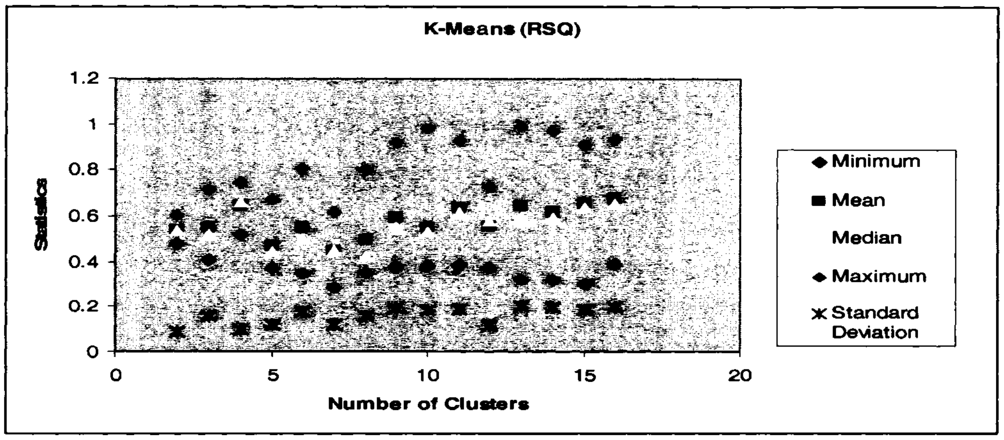
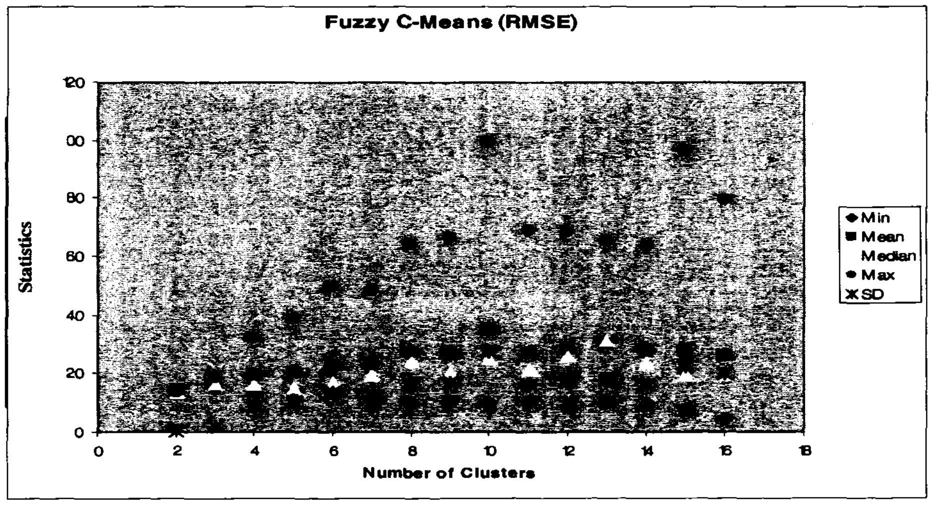
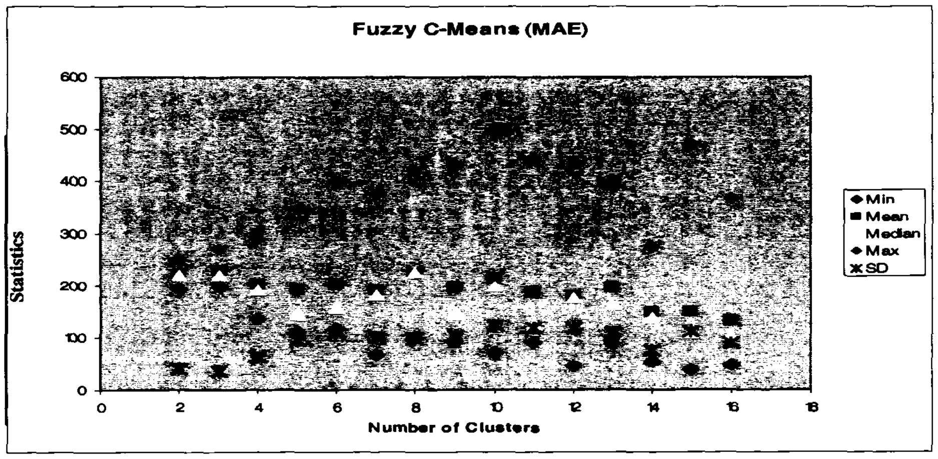


Figure 7.4. Variation of performance statistics among clusters (K- means)





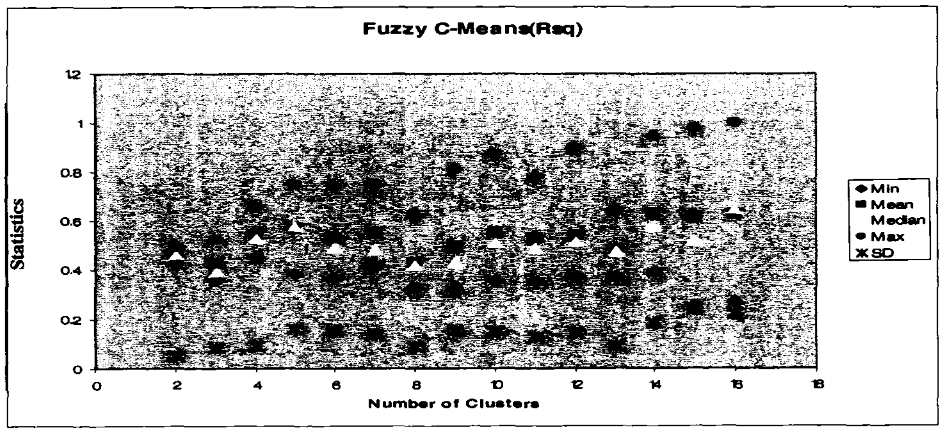


Figure 7.5. Variation of performance statistics among clusters (FCM)

To select the optimal number of clusters, the SVM methodology was trained and validated for each of the clustered datasets. Figure 7.4 and Figure 7.5 show the variation in the model performance measures (RMSE, MAE, and R^2) for the clustered datasets using the k-means and FCM clustering algorithms. An optimal number of clusters is desirable so that the model performance measures for all the clusters are high. Figure 7.5 for the FCM algorithm indicates that the model performance is best when five clusters were selected. Figure 7.6 shows the variation in key statistics for model performance measures in terms of R^2 for both the algorithms. The model performance on clusters generated by k-means is marginally better than the FCM clustering algorithm. Upon consideration of the spatial discreteness of the clusters and the SVM model performance on the clusters, the FCM methodology is selected for gold mineral reserve estimation in the study area. Figure 7.7 shows the location of the clusters for the optimal situation.

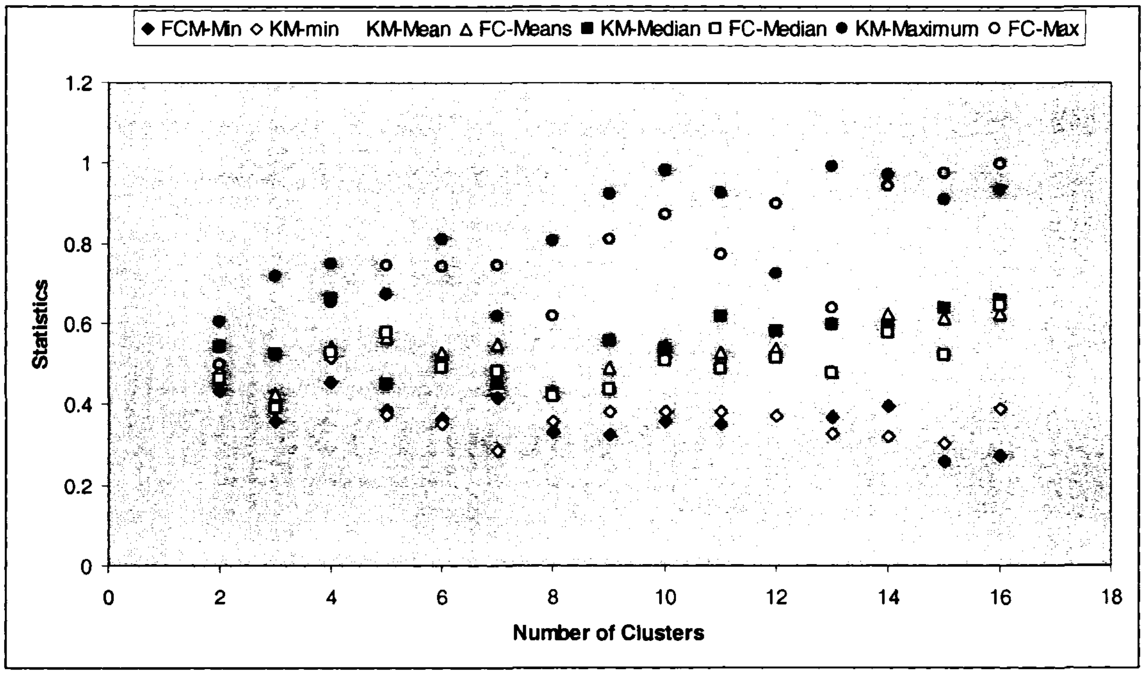


Figure 7.6. Variation in R^2 for the clusters created using the K-means and FCM

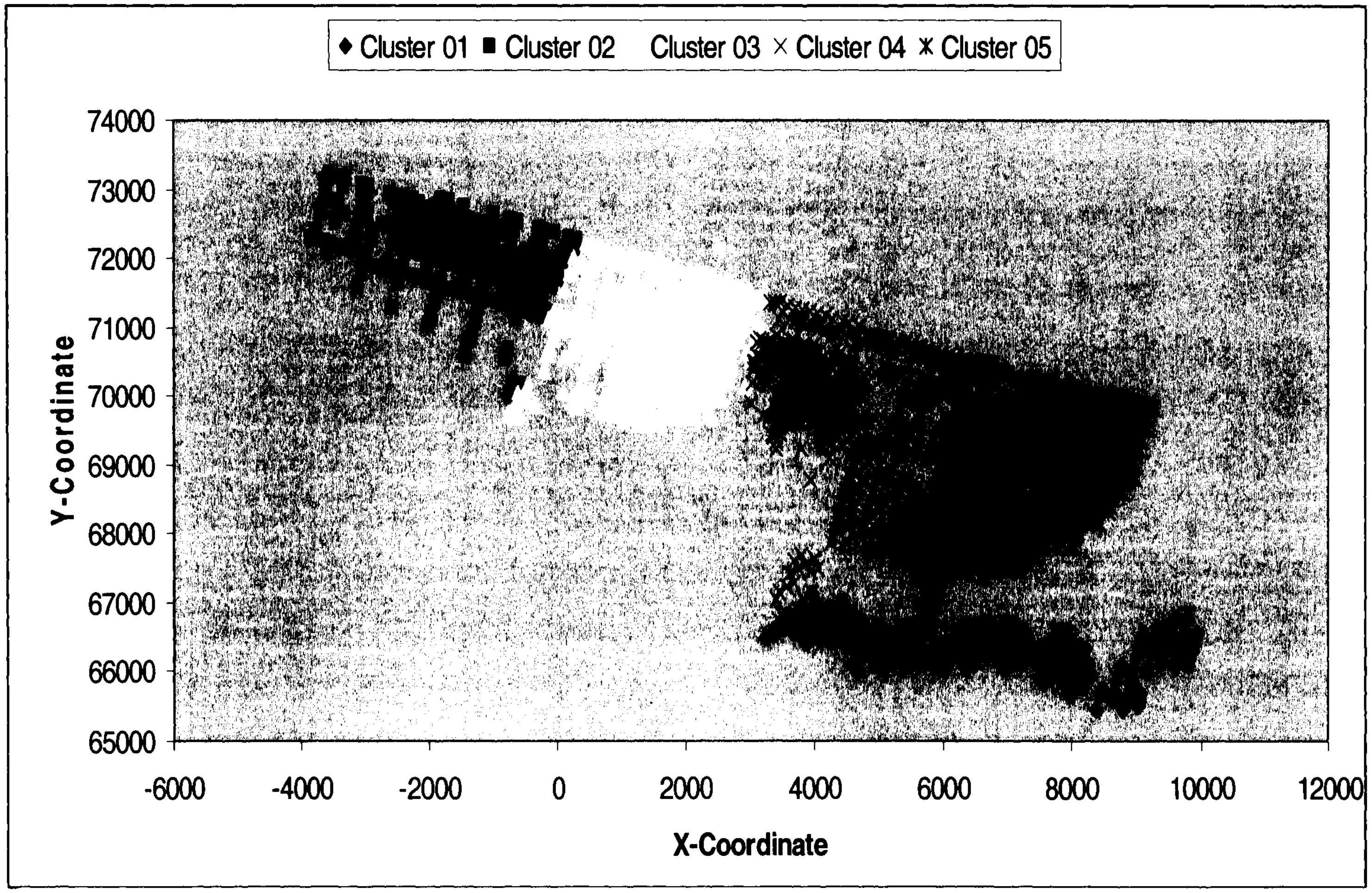


Figure 7.7. Map showing the location of the optimal number of clusters

Using five clusters for optimality, the FCM clustering algorithm was used to generate clusters. The SVM method was then applied to predict the gold grades on the datasets set aside and unseen during the process of training of the model for the development of the five clusters (Table 7.1). Figure 7.8 shows the observed and predicted gold grades for optimal clustering by the SVM method. Table 7.2 lists the SVM model performance measures for each of the five clusters. Unlike the results from the SVM application on the arbitrary fish blocks, the SVM performance is slightly improved and uniform for all the blocks.

Table 7.1: Number of samples in the cluster validation datasets

Mining Blocks	No. of Drill holes (validation dataset)
Cluster 1	97
Cluster 2	85
Cluster 3	91
Cluster 4	164
Cluster 5	154

Table 7.2: Model performance for the optimal clusters (FCM)

Cluster #	RMSE	ME	R^2
1	847.0	11.1	0.18
2	369.5	-52.5	0.27
3	363.5	-27.5	0.29
4	463.2	-40.0	0.26
5	562.1	-120.0	0.15

Presence of high gold grades and moderate-high gold grades in the study area are prerequisites for a successful mining venture. Figure 7.7 shows an important result of the FCM clustering application in mineral reserve estimation. The FCM clustering algorithm has shown great promise in delineating the zones of high grades within the study area. Based on the value of the maximum gold grade estimated, clusters that offer profitable mining conditions can be identified. Out of the five clusters generated, two clusters (cluster # 1 and 4) seem to contain high gold grades and probably represent mining blocks with high potential. Consequently, zones of poor grade are also identified (such as cluster # 5).

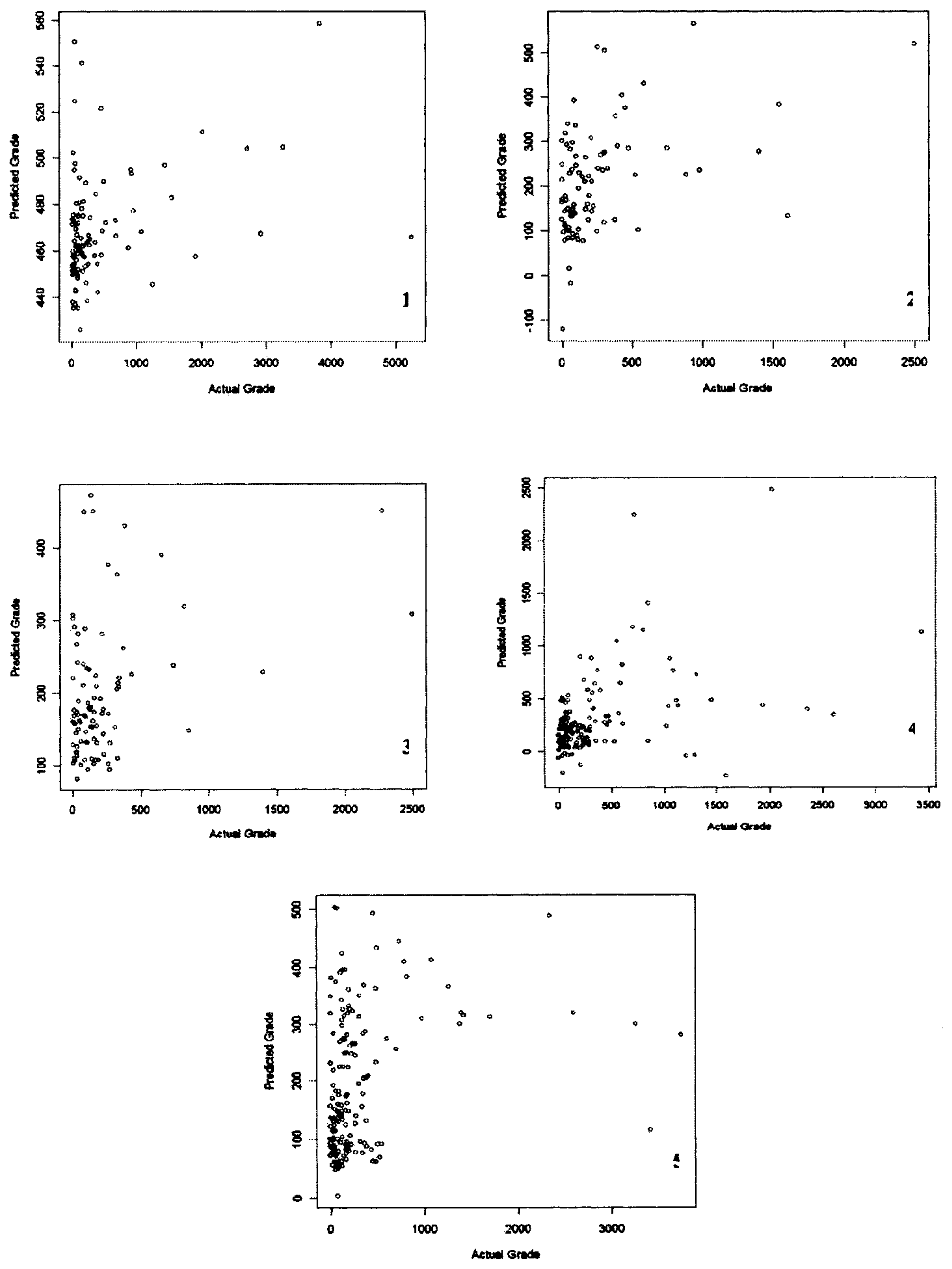


Figure 7.8. Predicted vs. observed gold grades for the clusters by FCM

7.2.1 Gold Reserve Estimation

After the clustering analysis, the gold grades are predicted throughout each of the clusters for a depth of 5m from the seafloor. For this purpose, the spatial extent of each of the clusters was divided into 25m × 25m grids and the SVM methodology was used to predict the gold grade in each of the grids. Figure 7.9 - Figure 7.13 show the predicted gold grades at each of the five clusters. It is of interest to note that most of the clusters with high grade potential are located in the eastern section of the study area.

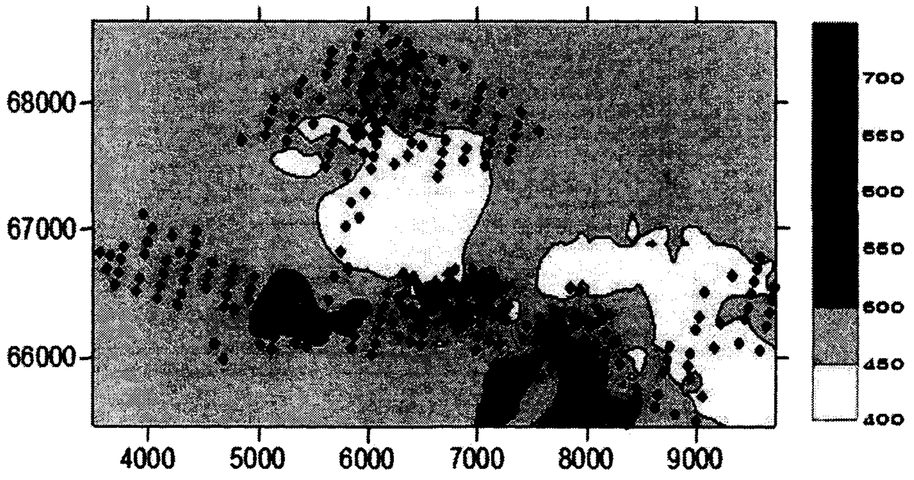


Figure 7.9. Predicted gold grade (in mg/ cu.m) in cluster # 1

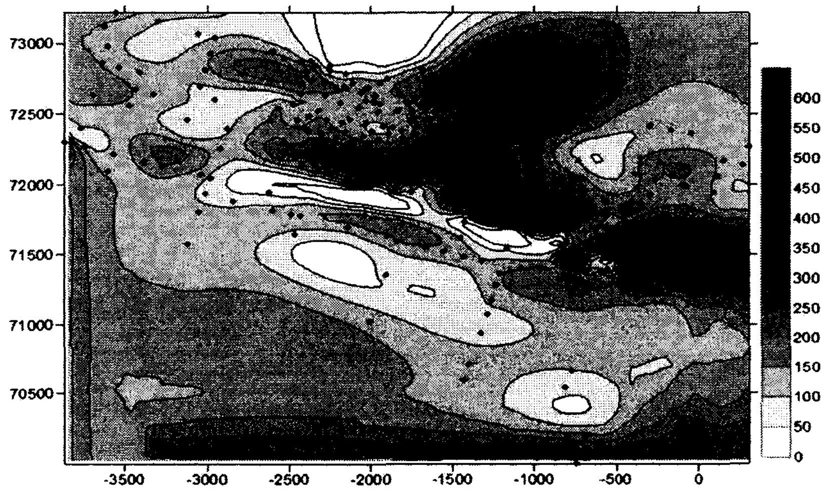


Figure 7.10. Predicted gold grade (in mg/ cu.m) in cluster # 2.

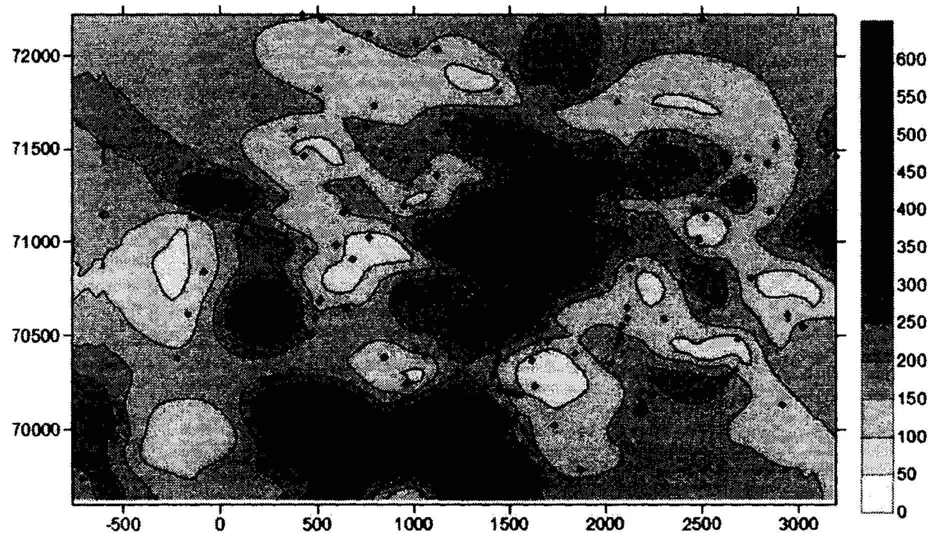


Figure 7.11. Predicted gold grade (in mg/ cu.m) in cluster # 3.

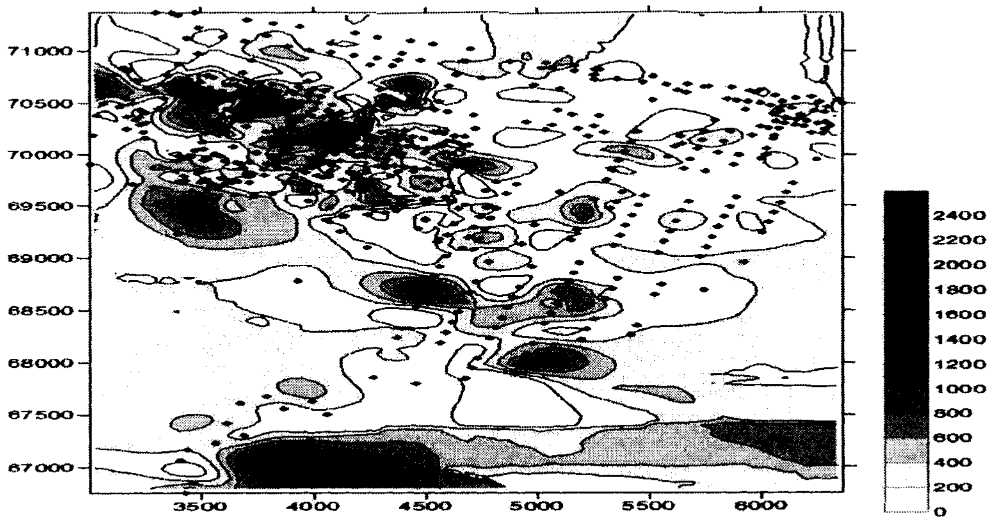


Figure 7.12: Predicted gold grade (in mg/cu.m) in cluster #4

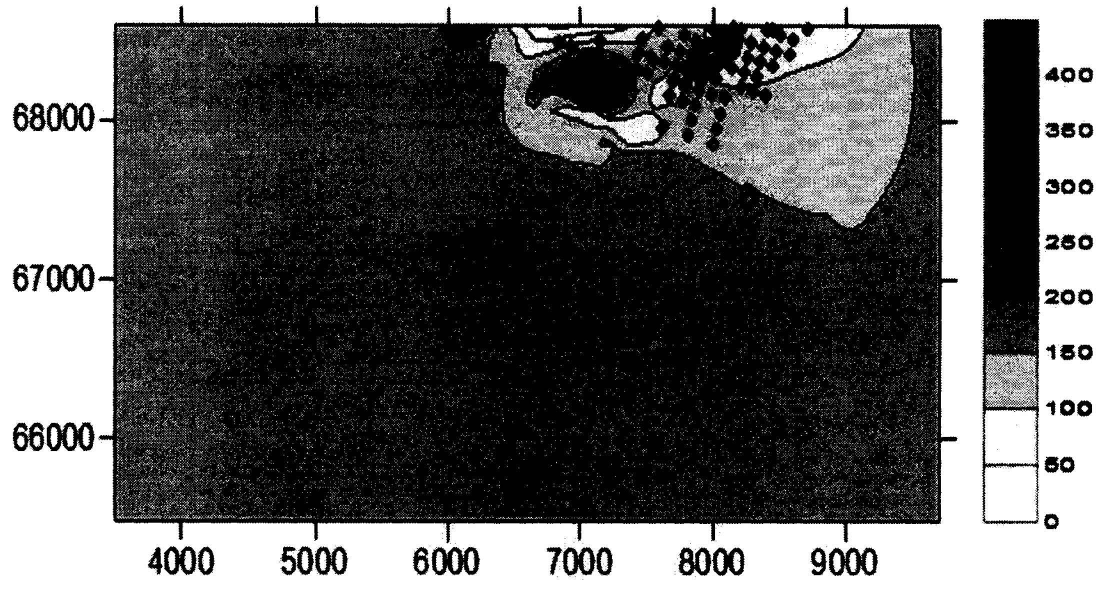


Figure 7.13. Predicted gold grade (in mg/ cu.m) in cluster # 5.

The predicted gold reserves for a depth of 5m in each of the clusters for different cutoff grades are presented in Table 7.3. The estimated gold grades indicate that clusters# 1 and 4 have high mining potential. Clusters# 2 and 3 have moderate potential while the cluster# 5 has the lowest potential. For selective mining, the density of gold grade is an important attribute. Therefore, the gold grade densities were calculated for each of the clusters. Based on the predicted density of gold grades, cluster #1 and #4 appear to have the most mining potential (Table 7.4). Figure 7.14 delineates the study area based on the gold mining potential.

Table 7.3. Estimated gold reserves for the cluster blocks

	Total reserve estimate (in tons) for different cut-off grades (COG, in mg/cu.m))				
Cluster#	COG300	COG400	COG500	COG600	COG700
1	19.45	19.45	2.05	0	0
2	1.21	0.624	0.237	0	0
3	1.48	0.685	0.105	0	0
4	6.21	5.03	3.90	3.05	2.51
5	0.06	0	0	0	0

Table 7.4. Estimated density of gold reserves for the cluster blocks

Cluster#	Density of gold reserves (tonnes/ Sq.km) for different cut-off grades (COG, in mg/cu.m))				
	COG300	COG400	COG500	COG600	COG700
1	2.34	2.34	0.25	0	0
2	0.16	0.086	0.033	0	0
3	0.20	0.091	0.014	0	0
4	0.73	0.592	0.458	0.358	0.295
5	0.007	0	0	0	0

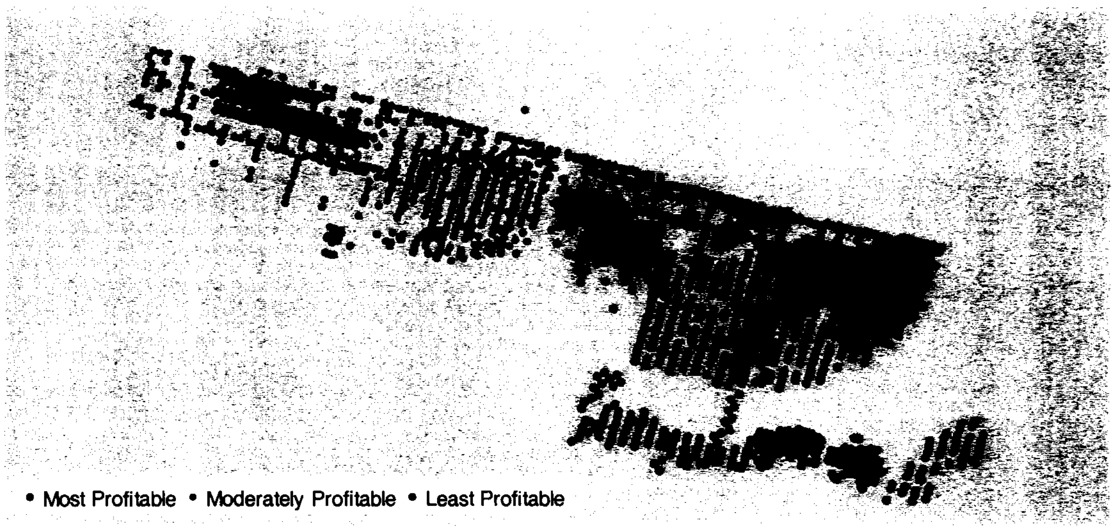

• Most Profitable • Moderately Profitable • Least Profitable

Figure 7.14. Map showing zones of varying potentiality for a gold mining scenario

The clustering methods, especially the FCM clustering algorithm, improved the performance of the SVM methodology over the entire dataset. The most important advantage of the clustering analysis is the ability to identify zones of high ore grade. Identification of high grade zones can be especially useful in the decision making process of future mining or exploration.

Chapter VIII

SUMMARY AND CONCLUSIONS

8.1 Summary

In this thesis, two machine learning algorithms, namely the neural network (NN) and support vector machine (SVM), were developed for the ore reserve estimation. The models were applied to a sparse and noisy drill hole dataset from the Nome gold district for the purpose of ore reserve estimation. Additionally, it was also applied on the Greens Creek dataset to estimate the silver content.

One of the characteristics of the Nome gold deposits is that the deposits were explored using few exploratory drill holes due to the prohibitive cost of sampling in the deposit. The sampling of the drill holes were further complicated due to high sea and general difficulties associated with undersea drilling and sampling. The problem was further compounded due to the depositional environment and sorting characteristics of the placer gold. As a result the data exhibits extreme spatial variations. In spite of these issues, attempts were made to treat the data judiciously, so that most of the information could be utilized for ore grade modeling. Several issues pertaining to the NN learning and model generalization were addressed. This included techniques for the development of the model data subsets (the training dataset, the calibration dataset and the validation dataset) and a study of three local optimization algorithms: (a) standard gradient descent back-propagation, (b) back-propagation with momentum algorithm (c) Levenberg-Marquardt backpropagation algorithm (LMBP).

The datasets which are used to develop a model play a critical role in its generalization ability and subsequently on the parameters that are obtained for ore reserve estimation. Conventional methods of randomly distributing the samples can result in statistically dissimilar data subsets, thereby reducing its reliability. In this regard, an ideal technique for legitimate data division has been suggested in the thesis. An integrated approach of data segmentation and genetic algorithms /kohonen network was investigated. The reason for data segmentation was primarily the presence of extreme high values in the data. Disproportionate representation of these extreme values in the subsets might result in dissimilar statistical characteristics. Therefore, data segmentation was carried out to divide the entire data into three segments such that each segment represented the low, the medium and the high grade gold values in equal proportion. Either genetic algorithms or kohonen networks were applied in each segment for sampling the data into the corresponding data subsets. The results indicated that a combination of data segmentation and genetic algorithms produced the best data division.

Several network architectures with different number of hidden layers and neurons in each layer were also investigated to determine the optimum network parameters. Once the behavior of the various optimization algorithms was identified and the network parameters obtained, they were used to develop the NN model. During the model development, it was trained on the training dataset and its performance was observed in a calibration dataset. The training was stopped, employing the quick stop method, which is aimed at improving the model generalization ability. The NN model was then used to estimate the ore grades associated in each of the fish block in the study area. Following the state lease pattern, the entire study area was divided into nine fish blocks (coho, halibut, herring, humpy, king, pink, red, silver and tomcod) for the purpose of gold reserve estimation. Apart from developing the NN model, a novel statistical learning based SVM approach was also considered. This approach tends to alleviate the data sparseness, impreciseness and the noise characteristics of the data.

For the development of a SVM based model, a calibration dataset is not required. So the training and the calibration datasets were merged during the SVM model development. A rigorous grid search employing the k-fold cross validation technique was carried out using the training dataset to select the optimal model parameters. Once the optimal parameters were selected, the SVM model was used for the estimation of the nome placer gold reserve.

To investigate the accuracy and promise of the SVM and NN as a tool for ore reserve estimation, the ore grade and the tonnage obtained from these machine learning algorithms is compared with those estimated by the predominantly used geostatistical ordinary kriging (OK) technique under various cut-off grades. The overall performance compared by the analysis of Mean Square Error (MSE), Mean Absolute Error (MAE), Mean Error (ME) and the coefficient of determination (R^2) indicated that the SVM model performance is better.

The improvement from the summary statistics stated above was however marginal and largely affected by the extreme spatial variations associated with the typical placer depositional characteristics of the Nome gold. Since the datasets used for the model development generally dictates the prediction, the generalization was poor and results were difficult to interpret. This problem is often observed when model comparison is based on only a single dataset. Therefore, in order to further establish the reliability of these models as a tool for ore reserve estimation, they were also applied to a dataset obtained from a lode deposit. The dataset obtained from Greens Creek ore deposit is devoid of the typical characteristics seen in the Nome placer gold deposit. An analysis similar to the Nome data was performed i.e. the ordinary kriging, neural network and the support vector machine were used to estimate the grade of the variable silver values associated with the deposit. A new performance statistic was suggested to compare the results. This summary statistic termed as the 'skill value' is an entirely subjective measurement. It considers the ME, MAE, RMSE equally and applies a scaling to the R^2. By doing so it enables one to rank the various models based on their performance such that a lower skill value implies a better model. In this way, different models can be assessed for their relevance to a particular task.

The initial studies on the Nome placer ore reserve were based on nine fish blocks. These blocks were obtained arbitrarily from the study area. From a mining point of view, these arbitrary fish blocks do not make much sense. The lease boundary could be divided based on the concentration of ore grade values and could be clustered differently. Therefore, clustering algorithms were applied in a separate study to divide the entire dataset into an optimal number of clusters where each cluster corresponded to a mining block. The goal behind performing this exercise was to obtain feasible mining blocks that are spatially discrete and do not overlap with one another.

8.2 Conclusions

This study investigated various modeling issues pertaining to the application of the machine learning algorithms viz. the neural network and support vector machine for ore grade estimation. They were applied to calculate the ore reserve for Nome placer gold deposit and the Greens Creek lode deposit. Based on the study results, the following specific conclusions are derived:

(i) The drill hole information for the placer gold deposit of the Nome district is very sparse; as a result, random division of data fails to achieve the desired statistical similarity among the subsets of data on which the NN model is built. A simulation study revealed that almost 25% of the time, random data divisions are bad.

(ii) Genetic algorithm and Kohonen network emerged as two promising tools for effective data division. When compared to the kohonen network, genetic algorithm resulted in superior data division.

(iii) Data segmentation resulted in improved performances of genetic algorithm and kohonen network in data division.

(iv) An integrated approach of data segmentation and genetic algorithm proved to be the best way for data divisions.

(v) LMBP algorithm performed best among the local optimization algorithms for all the data sets in terms of the speed of convergence and the error generated.

(vi) Training the network for large number of epoch's results in an overfitting of a model. Quick-stop training provides an efficient way to improve the model generalization; and thus a better solution.

(vii) Clustering algorithms generated five clusters as the optimum number of feasible mining blocks as compared to the nine fish blocks that were arbitrarily used.

(viii) The machine learning algorithms improved the generalization performance for most of the fish blocks in the Nome dataset except for the King and Tomcod blocks where the OK method generalized well. Among the two machine learning algorithms SVM produced the better results. The improvement was however marginal and it was largely affected by the extreme spatial variations associated with the typical placer gold depositional characteristics of the Nome gold.

(ix) Since the extreme variation in the Nome datasets heavily influenced the predictive capability of the three developed models and reduced their performance, their utility as a tool for reserve estimation could not be supported. So in order to further verify their ability for ore grade estimation they were used to predict the grade of silver values in a lode dataset. The SVM model produced the best estimates.

(x) Since the machine learning algorithms exhibited better performances it could be concluded, in general, that these methods can be used for the purpose of predictive mapping if the available data is used sensibly. Ore reserve estimation is one such predictive mapping. The choice of SVM and NN methods can also prove to be good since they are computationally fast compared to the OK method when the datasets are of small size. However, when the size of the dataset increases all of them are equally time consuming.

REFERENCES

An, P., Chung, C.F., and Rencz, A.N., (1995) "Digital lithology mapping from airborne geophysical and remote sensing data in the Melville peninsula, northern Canada, using a neural network approach", Remote Sensing of Environment, 53(2), pp76-84.

Bandopadhyay, S., Samanta, B., Ganguli, R., and Dutta, S., (2004) "Development of Marine Placer Reserve Model: Phase III- Investigation of Sparse Data Issues on Offshore Nome Gold Reserve Estimation": Working Report to Mineral Management Services, U.S Department of Interior.

Battiti, R., (1992) "First and Second order Methods for Learning: Between Steepest Descent and Newton's Method", Neural Computation, Vol. 4, No. 2, pp. 141-166.

Bishop, C.M., (1995) Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 482 pp.

Bowden, G.J., Maier, H.R., and Dandy, G.C., (2002) "Optimal Division of Data for Neural Network Models in Water Resources Application", Water Resources Research, Vol. 38, No. 2, 11 pp.

Boyce, M.S., Vernier, P.R., Nielson, S.E., and Schmiegelow, F.K.W., (2002) "Evaluating Resource Selection Functions", Ecological Modeling, Vol. 157, pp. 281-300.

Breiman, L., (1996) "Bagging Predictors", Machine Learning, Vol.26, No. 2, pp. 123-140.

Cardon, H.R.A., and Hoogstraten, R.V., (1995) "Key Issues For Successful Neural Network Applications: An Application in Geology", Artificial Neural Networks: an Introduction to ANN Theory and Practice, P. J. Braspenning, F. Thuijsman and A. J. M. M. Weijters eds., pp. 235-245.

Chang, C.C., and Lin, C., (2001), LIBSVM: A Library for Support Vector Machines. Available in: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cherkassky, V., and Ma, Y., (2002) "Practical Selection of SVM Parameters and Noise Estimation for SVM Regression", Neurocomputing, Special issue on SVM (Submitted).

Chung, F.L., and Lee, T., (1992), "Fuzzy Competitive Learning", Neural Networks, 7(3): 539-551.

Clarici, E., Owen, D. B., and Durucan, S., (1993) "Modeling and Representation of Spatial Data Using Artificial Neural Networks", Proceedings, Symposium on Innovative Mine Design for the 21st Century, pp. 199-204.

Cobb, E.H., (1974) "Placer Deposits of Alaska", Geological Survey Bulletin.

Collier, A. J., (1908) "Trillby Creek: Placer Deposits in the Nome Area", Massachusetts Institute of Technology, B.S. Thesis.

Cortes, C., and Vapnik, V., (1995) "Support Vector Networks", Machine Learning 20:273-297.

Denby, B., and Burnett, C., (1993) "A Neural Network Based Tool for Grade Estimation", 24th International Symposium on the Application of Computers and Operations Research in the Mineral Industries (APCOM), Montreal, Quebec.

Dixon, L.C.W., (1994) "Neural Networks and Unconstrained Optimization", in Algorithms for Continuous Optimization: The State of the Art (E. Spedicato, ed.), Netherlands: Kluwer Academic Publishers, pp. 513-530.

Dominy, S.C., Noppe, M.A., and Annels, A.E., (2002) "Errors and Uncertainty in Mineral Resource and Ore Reserve Estimation: The Importance of Getting it Right", Explor. Mining Geology, Vol. 11, No. 1-4, pp. 77-98.

Dubois, G., (2005) "European Report on Automatic Mapping Algorithms for Routine and Emergency Monitoring Data", Editor: G. Dubois, Publisher: Office for Official Publications of the European Communities, Luxembourg, ISBN 92-894-9400-X.

Dutta, S., Mishra, D., Ganguli, R., and Samanta, B., (2003) "Investigation of Two Neural Network Ensemble Methods for the Prediction of Bauxite Ore Deposit" Proceedings of the 6th International Conference on Information technology, CIT-2003. (Peer Reviewed).

Dutta, S., Ganguli, R., & Samanta, B., (2005a) "Investigation of Two Neural Network Methods in an Automatic Mapping Exercise", Journal of Applied GIS, Vol. 1, No. 2, August 2005.

Dutta, S., and Ganguli, R., (2005b) "Application of Boosting Algorithm in Neural Network Based Ash Measurement Using Online Ash Analyzers", 32nd International Symposium of the application of Computers and Operation Research in Mineral Industry, APCOM, 2005.

Dutta, S., Ganguli, R., and Samanta, B., (2005c) "Investigation of Two Neural Network Methods in an Automatic Mapping Exercise", in European Report on Automatic mapping algorithms for routine and emergency monitoring data. Report on the Spatial Interpolation Comparison (SIC2004) exercise, Editor: G. Dubois, Publisher: Office for Official Publications of the European Communities, Luxembourg, ISBN 92-894-9400-X.

Dutta, S., Ganguli, R., and Samanta, B., (2005d) "Comparative Evaluation of Radial Basis Functions and Kriging for Ore Grade Estimation", 32nd International Symposium of the application of Computers and Operation research in Mineral Industry, APCOM 2005, pp 203-211.

Dutta, S., Misra, D., Ganguli, R., Samanta, B. and Bandopadhyay, S., (2006a) "A Hybrid Ensemble Model of Kriging and Neural Network for Ore Grade Estimation", International Journal of Surface Mining, Reclamation and Environment, Vol.20,Number 1, March , pp 33-46.

Dutta, S., Bandopadhyay, S., and Samanta, B., (2006b) "Support Vector Machines for Ore Reserve Estimation", Mine Planning and Equipment Selection, MPES 2006 (Accepted).

Fitzgerald, R.W., and Lees, B.G., (1996) "Temporal Context in Floristic Classification", Computers and Geosciences, 22 (9), pp 981-994.

Foody, G.M., (1997) "Fully Fuzzy Supervised Classification of Land Cover from Remotely Sensed Imagery with an Artificial Neural Network", Neural Computing & Applications, 5 (4), pp 238-247.

Forgy, E.W., (1965) "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications", Biometrics, 21: 768-769.

Gallagher, M.R., (1999), Multi-layer Perceptron Error Surfaces: Visualization, Structure and Modeling, Unpublished PhD Dissertation, University of Queensland, 225p.

Ganguli, R., and Bandopadhyay, S. (2003) "Dealing with Sparse Data Issues in a Mineral Industry Neural Network Application", Proceedings, Computer Applications in the Mineral Industry (CAMI), Calgary, Alberta, Canada, 5p.

Geoeas: Geostatistical Environmental Assessment Software, Environmental Protection Agency, USA.

Goldberg, D.E., (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Pearson Education Asia Pte Ltd. 412 pp.

Goutte, C., (1997), "Note on Free Lunches and Cross-Validation," Neural Computation", 9, 1211-1215.

Hagan, M.T., Demuth, H.B. and Beale, M., (1995). Neural Network Design, PWS Publishing Company, Boston, 19 chapters.

Hartigan, J. A., and Wong, M. A., (1979) "A K-means Clustering Algorithm", Applied Statistics, 28: 100-108.

Hastie, T., Tibshirani, R., and Friedman, J., (2001) The Elements of Statistical Learning Theory- Data Mining, Inference and Prediction. Springer, New York.

Haykins, S., (1999). Neural Networks: A Comprehensive Foundation (2nd Ed): Prentice Hall, New Jersey, 824 pp.

Hodouin, D., Thibault, J. and Flament, F., (1991) "Artificial Neural Networks: An Emerging Technique to Model and Control Mineral Processing Plants", Proceedings, SME Annual Meeting, SME Preprint No. 91-148, 10pp.

Hopkins, D. M., and MacNeil, L., (1960) "Dredged Area: Alaska Division of Mines and Minerals".

Hoppner, F., Klawonn, F., Kruse, R., and Runkler, T., (1999) Fuzzy Cluster Analysis, John Wiley & Sons, ISBN. 0 471 98864 2

http://www.hecla-mining.com/propGreens.html

http://www.kennecottminerals.com/S&E_2002/GreensCreek.pdf

Isaaks, E.H., and Srivastava, R.M., (1989) Applied Geostatistics, Oxford University Press, ISBN 0-19-505012-6.

Journel, A. G., and Huijbregts, Ch. J., (1978), Mining Geostatistics, Academic Press.

Kanevski, M., Arutyunyan, R., Bolshov, L., and Maignan, M., (1996) "Artificial Neural Networks and Spatial Estimations of Chernobyl Fallout", Geoinformatics, 7(1), 5-11.

Kanevski, M., Pozdnoukhov, A., Canu, S., and Maignan, M., (2002) "Advanced Spatial Data Analysis and Modeling with Support Vector Machines", International Journal of Fuzzy Systems, March, 4(1), 606-616.

Kapageridis, I.K., and Denby, B., (1998) "Neural Network Modeling of Ore Grade Spatial Variability", Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN 98), Skovde, Sweden, pp. 209-214.

Ke, J., (2002) "Neural Network Modeling for Placer Ore Grade Spatial Variability", PhD Thesis, University of Alaska Fairbanks.

Kecman, V., (2000) Learning and Soft Computing: Support Vector Machines, Neural Network and Fuzzy Logic Models ISBN:0-262-11255-8, MIT publishers.

Kecman, V., (2004): Suport Vector Machines Basics- An Introduction Only", University of Aukland, School of Engineering Report, April.

Lee, C., and Sterling, R., (1992) "Identifying Probable Failure Modes for Underground Openings using a Neural Network", International Journal of Rock Mechanics, Mining Science and Geomechanics, Abstract., Vol. 29, No. 1, pp. 49-67.

Lippmann, R.P., (1987) "An Introduction to Computing with Neural Networks". IEEE Acoustics Speech and Signal Processing Magazine, Vol. 4 No. 2, pp. 4-22.

Mann, S., and Benwell, G. L., (1996) "The Integration of Ecological Neural and Spatial Modelling for Monitoring and Prediction for Semi-arid Landscapes", Computers and Geosciences, 22 (9), pp. 1003-1012.

Matlab-The Language of Technical Computing. The Mathworks Inc. Copyright 1984-2002.

Miller, D.M., Kaminsky, E.J., and Rana, S., (1995) "Neural Network Classification of Remote-Sensing Data", Computers & Geosciences, 21(3), pp. 377-386.

Miller, R.E. (2000) Optimization-Foundations and Applications, Wiley-Interscience publication, ISBN 0-471-32242-3.

Misra, D., Samanta, B., Dutta,S., and Bandopadhyay, S., (2005) "Evaluation of Artificial Neural Network and Kriging for the Prediction of Arsenic in Bedrock Derived Stream Sediments using Gold Concentration Data", International Journal of Surface Mining, Reclamation and Environment (under review).

Mukherjee, S., Osuna, E., and Girosi, F., (1997) "Nonlinear Prediction of Chaotic Time Series using a Support Vector Machine", In J. Principle, L. Gile, and E.Wilson, editors, Neural Networks for Signal Processing VII- Proceedings of the 1997 IEEE workshop.

NeuroShell 2, Release 4, Copyright 1993-2000 by Ward Systems Group$^{\oplus}$.

Pal, N.R., Bezdek, J.C., and Hathaway, R.J., (1996) "Sequential Competitive Learning and the Fuzzy c-Means Clustering Algorithms", Neural Networks, 9(5), 787-796.

Pan, G., (1995) "Practical Issues of Geostatiscal Reserve Estimation in the Mining Industry," CIM Bulletin, pp. 31-37.

Park, D.W., Deb, D., Jiang, Y.M., and Sanford, R.L., (1995) "Monitoring and Prediction of Longwall Instability using Artificial Intelligence Techniques", Proceedings, SME Preprint No. 95-147, 8pp.

Pozdnoukhov, A., (2005) "Support Vector Regression for Automated Robust Spatial Mapping of Natural Radioactivity", Journal of Applied GIS, Vol.1, No.2, August 2005.

Pozdnoukhov, A., Kanevski, M., Maignan, M., and Canu, S., (2002) "Robust Mapping of Spatial Data with Support Vector Regression" Preprint IBRAE-2002, Nuclear Safety Institute RAS, 2002, 15p.

R- version 2.1.1 (2005-06-20). Free Software Foundation, Inc., Copyright (C) 1989, ISBN 3-900051-09-7.

Rendu, J.M., (1979) "Kriging, Logarithmic Kriging, and Conditional Expectation: Comparison of Theory with Actual Results", Proceedings, 16th APCOM symposium, Tucson, Arizona, pp. 199-212.

Rizzo, D.M., and Doughetry, D.E., (1994) "Characterization of Aquifer Properties Using Artificial Neural Networks: Neural Kriging", Water Resources Research, Vol. 30, No. 2, pp. 483-497.

Rogers, L.L. And Dowla, F.U., (1994) "Optimization Of Groundwater Remediation Using Artificial Neural Networks With Parallel Solute Transport Modeling", Water Resources Research, Vol. 30, pp. 483-497.

Rusanowski, P.C., (1994) Nome Offshore Gold Placer Project: Nova Natural Resources Corp., Alaska, 292 p.

Samanta, B., Bandopadhyay, S., and Ganguli, R., (2004a) "Data Segmentation and Genetic Algorithms for Sparse Data Division in Nome Placer Gold Grade Estimation Using Neural Network and Geostatistics", Exploration and Mining Geology, Vol.11, pp. 69-76.

Samanta, B., Bandopadhyay, S., Ganguli, R., and Dutta, S., (2004b) "Sparse Data Division using Data Segmentation and Kohonen Network for Neural Network and Geostatistical Ore Grade Modeling in Nome Offshore Placer Deposit", Natural Resources Research, Vol. 13, No. 3, pp. 189-200.

Samanta, B., Bandopadhyay, S., Ganguli, R., and Dutta, S., (2005a)"Comparative Study of the Performance of Single Neural Network vs. Adaboost Algorithm Based Combination of Multiple Neural Networks for Mineral Resource Estimation", Journal of South African Institute of Mining and Metallurgy, Vol. 105, April.

Samanta, B., Ganguli, R., and Bandopadhyay, S., (2005b) "Comparing the Predictive Performance of Neural Networks with Ordinary Kriging in a Bauxite Deposit", Transactions of Institute of Mining and Metallurgy, Section A, UK, Vol.114, pp 129-139.

Samanta, B., Bandopadhyay, S., and Ganguli, R., (2006) "Comparing Evaluation of Neural Network Learning Algorithms for Ore Grade Estimation", Mathematical Geology, Vol. 38, No. 02, February.

Shang, Yi., and Wah, B. J., (1996), "Global Optimization for Neural Network Training", IEEE Computer, Vol. 29, no.3, pp.45-54.

Sharky, A.J.C., (1999) Combining Artificial Neural Nets- Ensemble and Modular Multi-Net Systems, Springer-Verlag Publication, ISBN-185233004X.

Smola, A. J., and Scholkopf, B., (1998) "A Tutorial on Support Vector Regression", NeuroCOLT2 Technical Report NC2-TR-1998-030.Available in: http://citeseer.csail.mit.edu/smola98tutorial.html.

Smola, A.J., and Scholkope, B., (2004) "A Tutorial on Support Vector Regression", Statistics and Computing 14, 199-222.

Surfer- Version 8.0, Golden Software, Inc., Copyright 2002.

Twarakavi, N.C., Misra, D., and Bandopadhyay, S., (2006) "Prediction of Arsenic in Bedrock Derived Stream Sediments at a Gold Mine Site under Conditions of Sparse Data", Natural Resources Research, May (In Print).

Twarakavi, N.C., Al Masari, M., Kaluarachchi, J.J., and Misra, D., (2006) "Design of Regional Scale Groundwater Quality Networks", Groundwater (In Review).

Tu, W., and Zhou, X., (1999) "A Wald Test Comparing Medical Costs Based on Log-Normal Distributions with Zero Valued Costs", Statistics in Medicine, Vol. 18, pp. 2749-2761.

Yama, B.R., and Lineberry, G.T., (1999) "Artificial Neural Network Application for a Predictive Task in Mining", Mining Engineering, February, pp. 59-64.

Yu, S., Ganguli, R., Walsh, D.E., Bandopadhyay, S., and Patil, S.L., (2003) "Calibration of On-line Analyzers Using Neural Networks", Mining Engineering, Vol. 56, No. 9,September, pp. 99-102.

Vapnik, V., (1995) The Nature of Statistical Learning Theory. Springer, New York.

Vapnik, V., (1998) Statistical Learning Theory. Wiley, New York.

Wu, X., and Zhou, Y., (1993) "Reserve Estimation using Neural Network Techniques", Computers & Geosciences, Vol. 19, No.4, pp. 567-575, Pergamon Press.

APPENDIX A

MATLAB CODE

****

Genetic Algorithm for Data Division

****

```
function sortdata=geneticfdv(data,ptr,pcal,pval,npop)
% THis function used the genetic algorithms for data division
xdata=data(1:end,1);
ydata=data(1:end,2);
wdata=data(1:end,3);
gdata=data(1:end,4);
dl=length(data);
mxdata=mean(xdata);
mydata=mean(ydata);
mwdata=mean(wdata);
mgdata=mean(gdata);
xstd=std(xdata);
ystd=std(ydata);
wstd=std(wdata);
gstd=std(gdata);
normx=(xdata-mxdata)./xstd;
normy=(ydata-mydata)./ystd;
normw=(wdata-mwdata)./wstd;
```

```
normg=(gdata-mgdata)./gstd;
normdata=[normx normy normw normg];
ntr=floor(0.01.*dl.*ptr);
ncal=floor(0.01.*dl.*pcal);
nval=dl-(ntr+ncal);
dset=[];
for i=1:npop
  ran=rand(dl,1);
   sran=sort(ran);
   for j=1:dl
       for k=1:dl
           if(sran(j)==ran(k))
               dset=[dset;k];
           end;
       end;
   end;
   datset(:,i)=dset;
   dset=[];
end;
parentdat=datset;
% Setting dfittest1 to a arbitary intial high value
dfittest1=1000;
bCondition=0;
```

```
while(bCondition==0)
for i=1:npop
f(i)=fitness(parentdat(:,i),normdata,ntr,ncal,nval,dl);
end;
dfittest2=min(f);
for i=1:npop
  if(f(i)==dfittest2)
    bestpos=i;
  end;
end;
if(dfittest1<=dfittest2)
    nConditionCount=nConditionCount+1;
        else
          nConditionCount=0;
          dfittest1=dfittest2
    bestsol=parentdat(:,bestpos);
  end;
  %check whether the changes in the fittest gene has occurred since 250
    if(nConditionCount >250)
    bCondition=1;
  end;
slct=select(f,npop);
childdat=generation(parentdat,slct,dl,npop);
```

```
for i=1:npop
   child=childdat(:,i);
   child=normalization(child,dl);
   childdat(:,i)=child;
end;
parentdat=childdat;
end;
bestsol;
sortdata=[];
for i=1:dl
 sortdata=[sortdata; data(bestsol(i),1) data(bestsol(i),2) data(bestsol(i),3)
data(bestsol(i),4)];
end;
mxdata=mean(xdata);
mydata=mean(ydata);
mwdata=mean(wdata);
mgdata=mean(gdata);
xstd=std(xdata);
ystd=std(ydata);
wstd=std(wdata);
gstd=std(gdata);
mxtr=mean(sortdata(1:ntr,1));
stdxtr=std(sortdata(1:ntr,1));
```

```
mxcal=mean(sortdata(ntr+1:ntr+ncal,1));
stdxcal=std(sortdata(ntr+1:ntr+ncal,1));
mxval=mean(sortdata(ntr+ncal+1:dl,1));
stdxval=std(sortdata(ntr+ncal+1:dl,1));
mytr=mean(sortdata(1:ntr,2));
stdytr=std(sortdata(1:ntr,2));
mycal=mean(sortdata(ntr+1:ntr+ncal,2));
stdycal=std(sortdata(ntr+1:ntr+ncal,2));
myval=mean(sortdata(ntr+ncal+1:dl,2));
stdyval=std(sortdata(ntr+ncal+1:dl,2));
mwtr=mean(sortdata(1:ntr,3));
stdwtr=std(sortdata(1:ntr,3));
mwcal=mean(sortdata(ntr+1:ntr+ncal,3));
stdwcal=std(sortdata(ntr+1:ntr+ncal,3));
mwval=mean(sortdata(ntr+ncal+1:dl,3));
stdwval=std(sortdata(ntr+ncal+1:dl,3));
mgtr=mean(sortdata(1:ntr,4));
stdgtr=std(sortdata(1:ntr,4));
mgcal=mean(sortdata(ntr+1:ntr+ncal,4));
stdgcal=std(sortdata(ntr+1:ntr+ncal,4));
mgval=mean(sortdata(ntr+ncal+1:dl,4));
stdgval=std(sortdata(ntr+ncal+1:dl,4));
ol=[mxdata mydata mwdata mgdata xstd ystd wstd gstd]
```

```
tr=[mxtr mytr mwtr mgtr stdxtr stdytr stdwtr stdgtr]

cal=[mxcal mycal mwcal mgcal stdxcal stdycal stdwcal stdgcal]

val=[mxval myval mwval mgval stdxval stdyval stdwval stdgval]

ntr; ncal; nval;
```

****

Function Fitness

****

```
function tdev=fitness(parentg,data,ntr,ncal,nval,dl)

% This function for assessing the fitness of the data division

for i=1:dl

   datax(i)=data(parentg(i),1);

   datay(i)=data(parentg(i),2);

   dataw(i)=data(parentg(i),3);

   datag(i)=data(parentg(i),4);

end;

mxtr=mean(datax(1:ntr));

mxcal=mean(datax(ntr+1:ntr+ncal));

mxval=mean(datax(ntr+ncal+1: ntr+ncal+nval));

varxtr=std(datax(1:ntr));

varxcal=std(datax(ntr+1:ntr+ncal));

varxval=std(datax(ntr+ncal+1: ntr+ncal+nval));

mytr=mean(datay(1:ntr));

mycal=mean(datay(ntr+1:ntr+ncal));

myval=mean(datay(ntr+ncal+1: ntr+ncal+nval));

varytr=std(datay(1:ntr));
```

```
varycal=std(datay(ntr+1:ntr+ncal));

varyval=std(datay(ntr+ncal+1: ntr+ncal+nval));

mwtr=mean(datay(1:ntr));

mwcal=mean(dataw(ntr+1:ntr+ncal));

mwval=mean(dataw(ntr+ncal+1: ntr+ncal+nval));

varwtr=std(dataw(1:ntr));

varwcal=std(dataw(ntr+1:ntr+ncal));

varwval=std(dataw(ntr+ncal+1: ntr+ncal+nval));

mgtr=mean(datag(1:ntr));

mgcal=mean(datag(ntr+1:ntr+ncal));

mgval=mean(datag(ntr+ncal+1: ntr+ncal+nval));

vargtr=std(datag(1:ntr));

vargcal=std(datag(ntr+1:ntr+ncal));

vargval=std(datag(ntr+ncal+1: ntr+ncal+nval));

mxdev=(mxtr-mxcal)^2+(mxcal-mxval)^2+(mxtr-mxval)^2;

mydev=(mytr-mycal)^2+(mycal-myval)^2+(mytr-myval)^2;

mwdev=(mwtr-mwcal)^2+(mwcal-mwval)^2+(mwtr-mwval)^2;

mgdev=(mgtr-mgcal)^2+(mgcal-mgval)^2+(mgtr-mgval)^2;

varxdev=abs(varxtr-varxcal)+abs(varxcal-varxval)+abs(varxtr-varxval);

varydev=abs(varytr-varycal)+abs(varycal-varyval)+abs(varytr-varyval);

varwdev=abs(varwtr-varwcal)+abs(varwcal-varwval)+abs(varwtr-varwval);

vargdev=abs(vargtr-vargcal)+abs(vargcal-vargval)+abs(vargtr-vargval);

tdev=mxdev+mydev+mwdev+mgdev+varxdev+varydev+varwdev+vargdev;
```

****

Function generation

****

```
unction child=generation(parent,slct,dl,npop)
% This function is for the production of population in the next generation
dpcrossover=.95;
dpmutate=.01;
j=1;
k=2;
while (k<=npop)
w=flip(dpcrossover);
if(w==1)
   ncross=ceil(rand(1,1).*dl);
else
   ncross=dl;
end;
p1=slct(j);
p2=slct(k);
[child1,child2]=crossover(parent(:,p1),parent(:,p2),ncross,dl);
child(:,j)=child1;
child(:,k)=child2;
w1=flip(dpmutate);
```

```
if(w1==1)
   ncross1=ceil(rand(1,1).*dl);
   ncross2=ceil(rand(1,1).*dl);
else
   ncross1=dl;
   ncross2=dl;
end;
[child(:,j),child(:,k)]=mutation(child(:,j),child(:,k),ncross1,ncross2,dl);
j=k+1;
k=j+1;
end;
```

****

Function Crossover

****

```
function[child1,child2]=crossover(parent1,parent2,ncross,dl)
% This function is for the crossover operation
if(ncross<dl)
for i=1:ncross
  child1(i,1)=parent1(i);
  child2(i,1)=parent2(i);


end;
for j=(ncross+1):dl
  child1(j,1)=parent2(j);
  child2(j,1)=parent1(j);


end;
else
  child1=parent1;
  child2=parent2;
end;
```

****

Function Mutation

****

```
function [child1,child2]=mutation(child1,child2,ncross1,ncross2,dl)
% This is mutation function
if((ncross1<dl)|(ncross2<dl))
  x=child1(ncross1);
  child1(ncross1)=child2(ncross2);
  child2(ncross2)=x;

else
   child1=child1;
   child2=child2;
end
```

****

Function Select

****

```
function slct=select(f,npop)

% This function is for the selection of random data division

for i=1:npop

invf(i)=1./f(i);

end;

suminvf=sum(invf);

for i=1:npop

   expected(i)=npop.*invf(i)./suminvf;

   %expected(i)=npop.*f(i)./sumf;

   jassign(i)=floor(expected(i));

   fraction(i)=expected(i)-jassign(i);

end;

k=0;

for i=1:npop

   while(jassign(i)>0)

      k=k+1;

      choices(k)=i;

      jassign(i)=jassign(i)-1;

   end;

end;
```

```
j=0;
while(k<npop)
  j=j+1;
  if (j>npop)
    j=1;
  end;
  if(fraction(j)>0)
    winner=flip(fraction(j));
    if(winner==1)
      k=k+1;
      choices(k)=j;
      fraction(j)=fraction(j)-1;
    end;
  end;
end;
   ran=rand(npop,1);
   sran=sort(ran);
   for i=1:npop
     for j=1:npop
       if(sran(i)==ran(j))
         slct(i)=choices(j);
       end;
     end;
```

```
end;
```

****

Function Flip

****

```
function winner=flip(dnum)
% This function is for generating 1 Or 0
dn=rand(1,1);
if(dnum>dn)
  winner=1;
else
  winner=0;
   end;
```

****

This function for the simulation study of data divisions

****

```
function simutest(data,nita,ptr,pc,pval)

% This function is for the testing of sparse data issues

dl=length(data);

ntr=ceil(0.01.*dl.*ptr);

ncal=ceil(0.01.*dl.*pc);

nval=dl-(ntr+ncal);

xdata=data(1:end,1);

ydata=data(1:end,2);

wdata=data(1:end,3);

gold=data(1:end,4);

xset=[];

yset=[];

wset=[];

gset=[];

nxt=0;

nyt=0;

nwt=0;

ngt=0;

nt=0;

for p=1:100
for i=1::nita
   ran=rand(dl,1);
   ran(1);
   sran=sort(ran);
   for j=1:dl
      for k=1:dl
         if(sran(j)==ran(k))
            xset=[xset; xdata(k)];
            yset=[yset; ydata(k)];
            wset=[wset; wdata(k)];
            gset=[gset;gold(k)];
         end;
      end;
   end;
   trxset=xset(1:ntr);
   calxset=xset(ntr+1:ntr+ncal);
   valxset=xset(ntr+ncal+1:dl);
   f1=anova(trxset,calxset,valxset);
   if(f1>3.0)
      nxt=nxt+1;
   end;
   tryset=yset(1:ntr);
   calyset=yset(ntr+1:ntr+ncal);
```

```
valyset=yset(ntr+ncal+1:dl);
f2=anova(tryset,calyset,valyset);
if(f2>3.0)
   nyt=nyt+1;
end;
trwset=wset(1:ntr);
calwset=wset(ntr+1:ntr+ncal);
valwset=wset(ntr+ncal+1:dl);
f3=anova(trwset,calwset,valwset);
if(f3>3.0)
   nwt=nwt+1;
end;
trgset=gset(1:ntr);
calgset=gset(ntr+1:ntr+ncal);
valgset=gset(ntr+ncal+1:dl);
w=waldtest(trgset,calgset,valgset);
mean(trgset);
if(w>5.99)
   ngt=ngt+1;
end;
if((f1>3.0)|(f2>3.0)|(w>5.99)|(f3>3.0))
   nt=nt+1;
end;
```

```
xset=[];
yset=[];
gset=[];
wset=[];
end;
end;
nxt=nxt./100
nyt=nyt./100
nwt=nwt./100
ngt=ngt./100
nt=nt./100
```

****

Function ANOVA

****

```
function f=anova(trdat,caldat,valdat)

% This function is for f test using anova

ntr=length(trdat);

ncal=length(caldat);

nval=length(valdat);

dl=ntr+ncal+nval;

mtr=mean(trdat);

mcal=mean(caldat);

mval=mean(valdat);

vartr=var(trdat);

varcal=var(caldat);

varval=var(valdat);

omean=(mtr+mcal+mval)./3;

msbs=((ntr.*(omean-mtr)^2)+(ncal.*(omean-mcal)^2)+(nval.*(omean-mval)^2))./2;

msws=(((ntr-1).*vartr)+((ncal-1).*varcal)+ ((nval-1).*varval))./(dl-3);

f=msbs./msws;
```

****

Function Waldtest

****

```
function w=waldtest(trgset,calgset,valgset)

% This function is for wald test of lognormal data

ntr=length(trgset);

ncal=length(calgset);

nval=length(valgset);

logtrgset=[];

logcalgset=[];

logvalgset=[];

for i=1:ntr

   if(trgset(i)~=0)

      logtrgset=[logtrgset; log(trgset(i))];

   end;

end;

llogtrgset=length(logtrgset);

mlogtrgset=mean(logtrgset);

varlogtrgset=var(logtrgset);

vtrgset=log(llogtrgset./ntr);

for j=1:ncal

   if(calgset(j)~=0)

      logcalgset=[logcalgset; log(calgset(j))];
```

```
  end;
end;
llogcalgset=length(logcalgset);
mlogcalgset=mean(logcalgset);
varlogcalgset=var(logcalgset);
vcalgset=log(llogcalgset./ncal);
for k=1:nval
  if(valgset(k)~=0)
    logvalgset=[logvalgset; log(valgset(k))];
  end;
end;
llogvalgset=length(logvalgset);
mlogvalgset=mean(logvalgset);
varlogvalgset=var(logvalgset);
vvalgset=log(llogvalgset./nval);
w1=(ncal.*exp(vcalgset).*(vtrgset+mlogtrgset+0.5.*varlogtrgset -
(vcalgset+mlogcalgset+0.5.*varlogcalgset))^2)./(1-
exp(vcalgset)+varlogcalgset+0.5*varlogcalgset^2)+
(nval.*exp(vvalgset).*(vtrgset+mlogtrgset+0.5.*varlogtrgset -
(vvalgset+mlogvalgset+0.5.*varlogvalgset))^2)./(1-
exp(vvalgset)+varlogvalgset+0.5*varlogvalgset^2);
w2=(((ntr*exp(vtrgset))*(1-exp(vtrgset)+varlogtrgset+0.5*varlogtrgset^2)^-1)+
((ncal*exp(vcalgset))*(1-exp(vcalgset)+varlogcalgset+0.5*varlogcalgset^2)^-1)+
((nval*exp(vvalgset))*(1-exp(vvalgset)+varlogvalgset+0.5*varlogvalgset^2)^-1))^-1;
```

```
w3=w1*w2;

w=w1-w3;
```

****

FUNCTION GRIDSEP

****

```
function [a,blc,gd]=gridsep(num)

% This function separates the grids insides the polygonal area from within

% the sampling region

load c:\blc.txt;

load c:\gd.txt;

a=[];

for i=1:size(blc,1)

if blc(i,3)==1.7e+038

a=[a;gd(i,:)];

end

end
```

##

```
*****************************************************************

        FUNCTION ESTMATION

*****************************************************************

function[est,res300,res400,res500,res600,res700,gr300,gr400,gr500,gr600,gr700]=
estimation(traindata,data,ninput,nhidden, noutput, w1, w2, b1, b2)

%This function esrimates for a particular cut-off grade.

input1=traindata(1:end,1);

input2=traindata(1:end,2);

input3=traindata(1:end,3);

output=traindata(1:end,4);

minin1=min(input1);

maxin1=max(input1);

minin2=min(input2);

maxin2=max(input2);

minin3=min(input3);

maxin3=max(input3);

mino=min(output);

maxo=max(output);

%"Data" is the Prediction data from the theoritical grid block

datain1=data(1:end,1);

datain2=data(1:end,2);

datain3=data(1:end,3);

nntrain1=(datain1-minin1)./(maxin1-minin1);
```

```
nntrain2=(datain2-minin2)./(maxin2-minin2);
nntrain3=(datain3-minin3)./(maxin3-minin3);
dl=length(data);
[estimation]=crossval(nntrain1,nntrain2,nntrain3,ninput,nhidden,noutput,w1,w2,b1,
b2,mino,maxo);
est=estimation';
res300=0;
res400=0;
res500=0;
res600=0;
res700=0;
z300=[];
gr300=[];
z400=[];
gr400=[];
z500=[];
gr500=[];
z600=[];
gr600=[];
z700=[];
gr700=[];
for i=1:size(est,1)
  if (est(i)<300)
```

```
      z300=[z300;est(i)];
    else
    gr300=[gr300;est(i)];
    res300=res300+est(i)*25*25*5;
    end
end
avggrade300=sum(gr300)/size(est,1);
totalreserve300=avggrade300*25*25*5*size(est,1);
for i=1:size(est,1)
    if (est(i)<400)
      z400=[z400;est(i)];
    else
    gr400=[gr400;est(i)];
    res400=res400+est(i)*25*25*5;
    end
end
avggrade400=sum(gr400)/size(est,1);
totalreserve400=avggrade400*25*25*5*size(est,1);
for i=1:size(est,1)
    if (est(i)<500)
      z500=[z500;est(i)];
    else
    gr500=[gr500;est(i)];
```

```
  res500=res500+est(i)*25*25*5;
  end
end
for i=1:size(est,1)
  if (est(i)<600)
   z600=[z600;est(i)];
  else
  gr600=[gr600;est(i)];
  res600=res600+est(i)*25*25*5;
  end
end
for i=1:size(est,1)
  if (est(i)<700)
   z700=[z700;est(i)];
  else
  gr700=[gr700;est(i)];
  res700=res700+est(i)*25*25*5;
  end
end
```

##

APPENDIX B

R CODE

****

```
#FIRST DATASET
library('class')
library('e1071')
train<-read.table('grp1_tr.txt')
xtrain=train[,1:3]
ytrain=train[,4]
xtrain1=xtrain
ytrain1=ytrain
xtrain1[,1]= (xtrain[,1]-min(xtrain[,1]))/(max(xtrain[,1])-min(xtrain[,1]))
xtrain1[,2]= (xtrain[,2]-min(xtrain[,2]))/(max(xtrain[,2])-min(xtrain[,2]))
xtrain1[,3]= (xtrain[,3]-min(xtrain[,3]))/(max(xtrain[,3])-min(xtrain[,3]))
ytrain1=(ytrain-min(ytrain))/(max(ytrain)-min(ytrain))

## Select the number of folds.
##attributes(model)----gives the list of attributes in the object.
a=1:7
b=1:7
for (cro in 1:7){
model <- svm(xtrain1, ytrain1,type="eps-regression",fitted=TRUE,cross=cro*2)
a[cro]=model$tot.MSE
b[cro]=cro*2
}
plot(b,a,xlab= "Number of Folds", ylab="MSE")
```

```
## TEST SET
test<-read.table('grp1_pr.txt')
xtest=test[,1:3]
ytest=test[,4]
xtest1=xtest
ytest1=ytest
xtest1[,1]= (xtest[,1]-min(xtrain[,1]))/(max(xtrain[,1])-min(xtrain[,1]))
xtest1[,2]= (xtest[,2]-min(xtrain[,2]))/(max(xtrain[,2])-min(xtrain[,2]))
xtest1[,3]= (xtest[,3]-min(xtrain[,3]))/(max(xtrain[,3])-min(xtrain[,3]))
ytest1=(ytest-min(ytrain))/(max(ytrain)-min(ytrain))


##OBJECTIVE TUNING

obj1<-tune(svm,xtrain1, ytrain1,ranges = list(gamma =seq(0.05,10, by=0.10),
cost=seq(1,15, by=0.5)),tunecontrol = tune.control(sampling="cross",cross=10),
type="eps-regression",kernel="radial",epsilon=0.05,scale=1)

plot(obj1)


#obj1<-tune(svm,xtrain1, ytrain1,validation.x=xtest1,validation.y=ytest1, ranges =
list(gamma =seq(0.05,9, by=0.10), cost=seq(1,10, by=0.5)),tunecontrol =
tune.control(sampling="fix",fix=1), type="eps-
regression",kernel="radial",epsilon=0.05,scale=1)

plot(obj1)
```

```
## SELECTING EPSILON
 k=1
 a=1:75
 b=1:75
 for (eps in 1:75){
 model <- svm(xtrain1, ytrain1,type="eps-
regression",cross=10,gamma=4.5,cost=.01,epsilon=eps/100)
 result<-predict(model,xtrain1)
 a[k]=sum((result-ytrain1)^2)/length(ytest1)
 b[k]=eps/100
 k=k+1
 }
 plot(b,a,xlab="epsilon",ylab="MSE")

## RUN THE MODEL
model<-svm(xtrain1,ytrain1,type="eps-
regression",kernel="radial",cross=10,cost=0.03,epsilon=0.5,gamma=4.55)

## TEST DATASET RESULTS
result_test<-predict(model,xtest1)
yresult_test=result_test*(max(ytrain)-min(ytrain)) +min(ytrain)
sq_error=sum((yresult_test-ytest)^2)
rmse=sqrt(sq_error/length(ytest))
mse=(sq_error/length(ytest))
me=sum((yresult_test-ytest))/(length(ytest))
mae=sum(abs(yresult_test-ytest))/(length(ytest))
test_est<-cbind(ytest,yresult_test)
```

```
library('boot')
r=corr(test_est, w=rep(1,nrow(test_est))/nrow(test_est))
rsq=r^2
rmse
mse
mse
me
mae
rsq

write.table(test_est,file="c:/test_est_grp1.txt",row.names=FALSE,
col.names=FALSE)

plot(ytest,yresult_test, xlab= "Actual Grade", ylab="Predicted Grade")

## TRAINING DATASET RESULT

result_tr<-predict(model,xtrain1)
ytr_pred = result_tr*(max(ytrain)-min(ytrain)) +min(ytrain)
sq_error=sum((ytr_pred-ytrain)^2)
rmse=sqrt(sq_error/length(ytrain))
mse=(sq_error/length(ytrain))
me=sum((ytr_pred-ytrain))/(length(ytrain))
mae=sum(abs(ytr_pred-ytrain))/(length(ytrain))
tr_est<-cbind(ytrain,ytr_pred)
library('boot')
r=corr(tr_est, w=rep(1,nrow(tr_est))/nrow(tr_est))
rsq=r^2
rmse
```

```
mse
me
mae
rsq

plot(ytrain, ytr_pred, xlab= "Actual Grade", ylab="Predicted Grade")
write.table(tr_est,file="c:/tr_est_grp1.txt",row.names=FALSE,
col.names=FALSE)

remove(xtrain1,ytrain1, xtrain, ytrain,
result_tr,obj1,train,model,xtest,xtest1,result_test,ytest,yresult_test,ytr_pred,ytr_pred)

attributes(object name)
```

R Code for Clustering Algoritms

##K-means Clustering

```
op <- par(mfrow = c(5,3), mar = .1+ c(2,2,3,1))
rmsesum=mat.or.vec(15,5)
abssum=mat.or.vec(15,5)
r2sum=mat.or.vec(15,5)
number=mat.or.vec(16, 16)

for (n in 2:16) {
        rm(cl,alldata,data1,data2,x,y,m,new, error)
        alldata<-read.table('data.txt')
        (cl <- kmeans(alldata, n, nstart = 25))
        plot(alldata[,1],alldata[,2], col = cl$cluster,xlab="X-coordinate", ylab="Y-
coordinate",main=paste("Number of Clusters=",n,sep=" "),cex=0.3, pch=0.35);
         alldata[,5]=cl$cluster
        number[1:n,n-1]=cl$size
        error=mat.or.vec(n,3)

if (n==2){
test<-write.table(alldata,"clustered2.txt",row.names = FALSE,col.names = FALSE)}
if (n==3){
test<-write.table(alldata,"clustered3.txt",row.names = FALSE,col.names = FALSE)}
if (n==4){
test<-write.table(alldata,"clustered4.txt",row.names = FALSE,col.names = FALSE)}
if (n==5){
test<-write.table(alldata,"clustered5.txt",row.names = FALSE,col.names = FALSE)}
if (n==6){
```

```
test<-write.table(alldata,"clustered6.txt",row.names = FALSE,col.names = FALSE)}
if (n==7){
test<-write.table(alldata,"clustered7.txt",row.names = FALSE,col.names = FALSE)}
if (n==8){
test<-write.table(alldata,"clustered8.txt",row.names = FALSE,col.names = FALSE)}
if (n==9){
test<-write.table(alldata,"clustered9.txt",row.names = FALSE,col.names = FALSE)}
if (n==10){
test<-write.table(alldata,"clustered10.txt",row.names = FALSE,col.names = FALSE)}
if (n==11){
test<-write.table(alldata,"clustered11.txt",row.names = FALSE,col.names = FALSE)}
if (n==12){
test<-write.table(alldata,"clustered12.txt",row.names = FALSE,col.names = FALSE)}
if (n==13){
test<-write.table(alldata,"clustered13.txt",row.names = FALSE,col.names = FALSE)}
if (n==14){
test<-write.table(alldata,"clustered14.txt",row.names = FALSE,col.names = FALSE)}
if (n==15){
test<-write.table(alldata,"clustered15.txt",row.names = FALSE,col.names = FALSE)}
if (n==16){
test<-write.table(alldata,"clustered16.txt",row.names = FALSE,col.names = FALSE)}

for (i in 1:n) {
                data1<-alldata[which(alldata$V5==i),]
                data2<-subset(data1,select=-V5)
                attach(data2)
                x <- subset(data2, select = -V4)
                y <- V4
obj <- tune(svm, x,y,ranges = list(gamma =seq(0.05,3, by=0.2), cost=seq(1,12,
by=.5)),tunecontrol = tune.control(sampling = "cross",cross=5))
                m <- svm(x, y,cost=obj$best.parameters[2],
gamma=obj$best.parameters[2], epsilon=0.05)
                new <- predict(m, x)
                error[i,1]= sqrt(sum((y-new)^2))/length(y)
                error[i,2]=sum(abs(y-new))/length(y)
                error[i,3]=(length(y)*sum(y*new)-
sum(y)*sum(new))/(sqrt(length(y)*sum(y*y)-
(sum(y)^2))*sqrt(length(y)*sum(new*new)-(sum(new)^2)))

                rmsesum[n-1,1]=min(error[,1])
                rmsesum[n-1,2]=mean(error[,1])
                rmsesum[n-1,3]=median(error[,1])
                rmsesum[n-1,4]=max(error[,1])
                rmsesum[n-1,5]=sd(error[,1])

                abssum[n-1,1]=min(error[,2])
                abssum[n-1,2]=mean(error[,2])
                abssum[n-1,3]=median(error[,2])
                abssum[n-1,4]=max(error[,2])
                abssum[n-1,5]=sd(error[,2])

                r2sum[n-1,1]=min(error[,3])
                r2sum[n-1,2]=mean(error[,3])
                r2sum[n-1,3]=median(error[,3])
                r2sum[n-1,4]=max(error[,3])
                r2sum[n-1,5]=sd(error[,3])
        }}
```

```
### write all the errors and rsq values to the workspace.
test1<-write.table(rmsesum,"rmsesum_km.txt",row.names = FALSE,col.names =
FALSE)
test2<-write.table(abssum,"abssum_km.txt",row.names = FALSE,col.names = FALSE)
test3<-write.table(r2sum,"r2sum_km.txt",row.names = FALSE,col.names = FALSE)
test4<-write.table(number,"kmeanclustersize.txt",row.names = FALSE,col.names =
FALSE)
```

##Fuzzy C-means clustering

```
op <- par(mfrow = c(5,3), mar = .1+ c(2,2,3,1))
fuzrmsesum=mat.or.vec(15,5)
fuzabssum=mat.or.vec(15,5)
fuzr2sum=mat.or.vec(15,5)
number=mat.or.vec(16, 16)

for (n in 2:16) {
        rm(cl,alldata,data1,data2,x,y,m,new, error)
        library('e1071')
        alldata<-read.table('data.txt')
        (cl <- cmeans(alldata, n, iter.max=150))
        number[1:n,n-1]=sort(cl$size,decreasing=TRUE)
        plot(alldata[,1],alldata[,2], col = cl$cluster,xlab="X-coordinate", ylab="Y-
coordinate",main=paste("Number of Clusters=",n,sep=" "),cex=0.3, pch=0.35)
        alldata[,5]=cl$cluster
        error=mat.or.vec(n,3)
```

```
if(n==2){
test<-write.table(alldata,"clustered2.txt",row.names = FALSE,col.names = FALSE)}
if(n==3){
test<-write.table(alldata,"clustered3.txt",row.names = FALSE,col.names = FALSE)}
if(n==4){
test<-write.table(alldata,"clustered4.txt",row.names = FALSE,col.names = FALSE)}
if(n==5){
test<-write.table(alldata,"clustered5.txt",row.names = FALSE,col.names = FALSE)}
if(n==6){
test<-write.table(alldata,"clustered6.txt",row.names = FALSE,col.names = FALSE)}
if(n==7){
test<-write.table(alldata,"clustered7.txt",row.names = FALSE,col.names = FALSE)}
if(n==8){
test<-write.table(alldata,"clustered8.txt",row.names = FALSE,col.names = FALSE)}
if(n==9){
test<-write.table(alldata,"clustered9.txt",row.names = FALSE,col.names = FALSE)}
if(n==10){
test<-write.table(alldata,"clustered10.txt",row.names = FALSE,col.names = FALSE)}
if(n==11){
test<-write.table(alldata,"clustered11.txt",row.names = FALSE,col.names = FALSE)}
if(n==12){
test<-write.table(alldata,"clustered12.txt",row.names = FALSE,col.names = FALSE)}
if(n==13){
test<-write.table(alldata,"clustered13.txt",row.names = FALSE,col.names = FALSE)}
if(n==14){
test<-write.table(alldata,"clustered14.txt",row.names = FALSE,col.names = FALSE)}
if(n==15){
test<-write.table(alldata,"clustered15.txt",row.names = FALSE,col.names = FALSE)}
```

```
if (n==16){
test<-write.table(alldata,"clustered16.txt",row.names = FALSE,col.names = FALSE)}

for (i in 1:n) {
                data1<-alldata[which(alldata$V5==i),]
                data2<-subset(data1,select=-V5)
                attach(data2)
                x <- subset(data2, select = -V4)
                y <- V4
                obj <- tune(svm, x,y,ranges = list(gamma =seq(0.05,3, by=0.2),
cost=seq(1,12, by=.5)),tunecontrol = tune.control(sampling = "cross",cross=5))
                m <- svm(x, y,cost=obj$best.parameters[2],
gamma=obj$best.parameters[2], epsilon=0.05)
                new <- predict(m, x)
                error[i,1]= sqrt(sum((y-new)^2))/length(y)
                error[i,2]= sum(abs(y-new))/length(y)
                error[i,3]=(length(y)*sum(y*new)-
sum(y)*sum(new))/(sqrt(length(y)*sum(y*y)-
(sum(y)^2))*sqrt(length(y)*sum(new*new)-(sum(new)^2)))

                fuzrmsesum[n-1,1]=min(error[,1])
                fuzrmsesum[n-1,2]=mean(error[,1])
                fuzrmsesum[n-1,3]=median(error[,1])
                fuzrmsesum[n-1,4]=max(error[,1])
                fuzrmsesum[n-1,5]=sd(error[,1])
                fuzabssum[n-1,1]=min(error[,2])
                fuzabssum[n-1,2]=mean(error[,2])
                fuzabssum[n-1,3]=median(error[,2])
                fuzabssum[n-1,4]=max(error[,2])
                fuzabssum[n-1,5]=sd(error[,2])
                fuzr2sum[n-1,1]=min(error[,3])
                fuzr2sum[n-1,2]=mean(error[,3])
                fuzr2sum[n-1,3]=median(error[,3])
                fuzr2sum[n-1,4]=max(error[,3])
                fuzr2sum[n-1,5]=sd(error[,3])
        }
}

###SRIDHAR - write all the errors and rsq values to the workspace.
test1<-write.table(fuzrmsesum,"fuzrmsesum_km.txt",row.names = FALSE,col.names =
FALSE)
test2<-write.table(fuzabssum,"fuzabssum_km.txt",row.names = FALSE,col.names =
FALSE)
test3<-write.table(fuzr2sum,"fuzr2sum_km.txt",row.names = FALSE,col.names =
FALSE)
test4<-write.table(number,"fuzclustersize.txt",row.names = FALSE,col.names = FALSE)

##optimal number of clusters for Fuzzy c-means

op <- par(mfrow = c(3,3))
fuzrmsesum=mat.or.vec(15,5)
fuzabssum=mat.or.vec(15,5)
fuzr2sum=mat.or.vec(15,5)
gamma1=c(6.5,8.0,8.0,9.0,12.0,
number=mat.or.vec(16, 16)
r2=mat.or.vec(9, 3)
```

```
n=9 #Optimal number of clusters
rm(cl,alldata,data1,data2,x,y,m,new, error)
library('e1071')
alldata<-read.table('data.txt')
(cl <- cmeans(alldata, n, iter.max=150))
number[1:n,n-1]=sort(cl$size,decreasing=TRUE)

alldata[,5]=cl$cluster
error=mat.or.vec(n,3)
for (i in 1:n) {
        data1<-alldata[which(alldata$V5==i),]
        data2<-subset(data1,select=-V5)
        attach(data2)
        x <- subset(data2, select = -V4)
        y <- V4
        #obj <- tune(svm, V4~., data = iris,ranges = list(gamma = 1:100/100, cost =
1:10),tunecontrol = tune.control(sampling = "cross",cross=5))
        m <- svm(x, y)    #, cost=obj$best.parameters[2],
gamma=obj$best.parameters[2], epsilon=0.05)
        new <- predict(m, x)

        plot(y,new, xlab="Observed Grade (mg/Cu.m)", ylab="Predicted Grade
(mg/Cu.m)", xlim=c(min(y),max(y)), ylim=c(min(y),max(y)), cex=0.3, pch=0.35)
        r2[i,1]=sqrt(sum((y-new)^2))/length(y)
        r2[i,2]=sum(abs(y-new))/length(y)
        r2[i,3]=(length(y)*sum(y*new)-sum(y)*sum(new))/(sqrt(length(y)*sum(y*y)-
(sum(y)^2))*sqrt(length(y)*sum(new*new)-(sum(new)^2)))}
```